John Knoll: Industrial Light & Magic's Graphics Magician

#287 JULY 1998

Dr. Dobb's JOURNAL

SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

http://www.ddj.com

GRAPHICS PROGRAMMING

- Creating Reactive Animations
- Inside the Kernel Graphics Interface
- A 3D Model Viewer for OpenGL

Your Own DVD-Video Player!

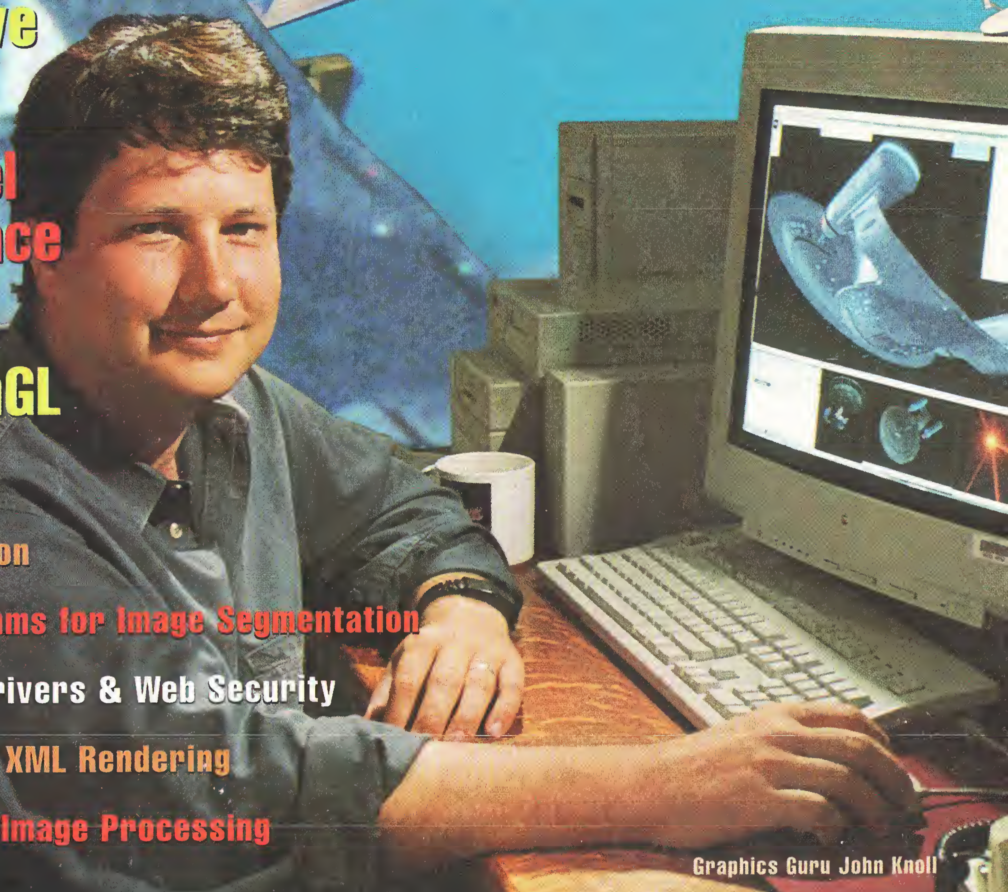
Off-the-Shelf Speech Recognition

Algorithms for Image Segmentation

JDBC Drivers & Web Security

XSL for XML Rendering

Java & Image Processing



Graphics Guru John Knoll

$4.95 ($5.95 CANADA)

07>

0 70992 35566 2

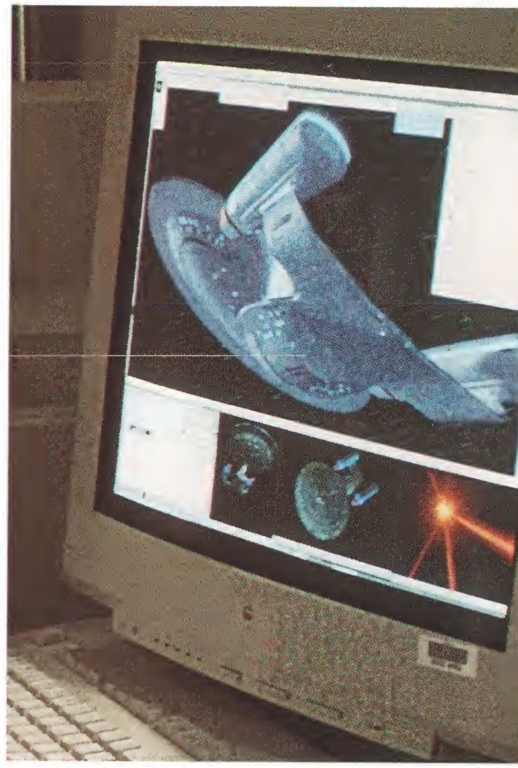
A Miller Freeman Publication

Dr. Dobb's JOURNAL

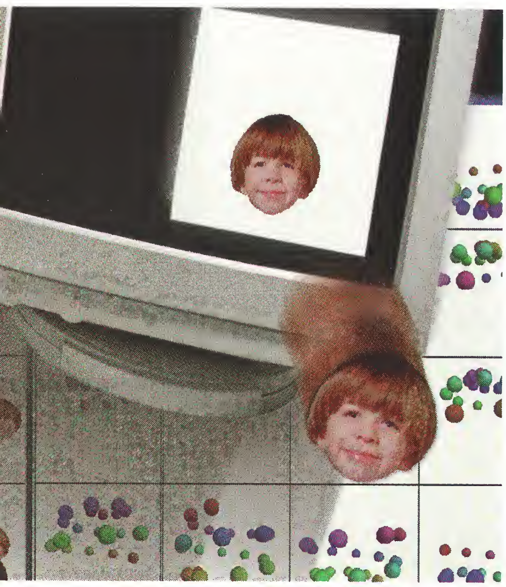
SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

CONT

FEATURES



Photography by Sean Casey courtesy of Industrial Light & Magic.



EMBEDDED SYSTEMS

INTERNET PROGRAMMING

ENTS

JULY 1998
VOLUME 23, ISSUE 7

PROGRAMMER'S TOOLCHEST

COLUMNS

FORUM

RESOURCE CENTER

As a service to our readers, source code (and related files), back-referenced articles, and relevant links are available electronically at this month's online Table of Contents at http://www.ddj.com/. Source code is also available via anonymous FTP from ftp.ddj.com (199.125.85.76), the DDJ Forum on CompuServe (type GO DDJ), and DDJ Online (650-358-8857, 14.4 kbps, 8-N-1). Source-code diskettes can be ordered ($14.95, California residents add sales tax) by mail, fax (650-358-9749), or phone (650-655-4100 x5701). Letters to the editor and article proposals/submissions should be mailed or faxed to the *DDJ* office or sent electronically to editors@ddj.com. Author guidelines are available at http://www.ddj.com/. Send inquiries or requests to *Dr. Dobb's Journal*, 411 Borel Ave., San Mateo, CA 94402. For subscription questions (including change of address), call 800-456-1215 (U.S. and Canada); other countries, call 303-678-0439 or fax 303-661-1885. E-mail subscription questions to 71572.341@compuserve.com or write to *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80322-6188.

NEXT MONTH

August brings our annual C++ programming issue.

Dr. Dobb's JOURNAL
SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

PUBLISHER *Peter Westerman*

EDITORIAL

EDITOR-IN-CHIEF *Jonathan Erickson*
MANAGING EDITOR *Deirdre Blake*
MANAGING EDITOR, DIGITAL MEDIA *Kevin Carlson*
SENIOR TECHNICAL EDITOR *Tim Kientzle*
TECHNICAL EDITOR *Eugene Eric Kim*
SENIOR PRODUCTION EDITOR *Monica E. Berg*
ASSOCIATE MANAGING EDITOR *Amy W. Wong*
ART DIRECTOR *Margaret A. Anderson*
CONTRIBUTING EDITORS *Al Stevens, Tom Genereaux, Scot Wingo, George Shepherd, Robert Collins, Cliff Berg, Steve Alexander, David Betz, Bruce Schneier, Mark Russinovich, Bryce Cogswell, Ray Duncan, Jack Woehr, Jon Bentley, Dennis Shasha*
EDITOR-AT-LARGE *Michael Swaine*
PRODUCTION MANAGER *Denise Denis*

CIRCULATION

DIRECTOR OF CIRCULATION *Jerry Okabe*
GROUP CIRCULATION MANAGER *Michael Poplardo*

MARKETING/ADVERTISING

SALES DIRECTOR, EAST *Brenner Fuller*
SALES DIRECTOR, WEST *Paul Miller*
MARKETING MANAGER *Holly Vessichelli*
PUBLISHER'S ASSISTANT *Jodie Medeiros*
PRODUCTION COORDINATOR *Michael Calderon*
ACCOUNT MANAGERS see page 128
GRAPHIC DESIGNER *Carey Perez*

MILLER FREEMAN INC.

CEO *Tony Tillin*
CHAIRMAN *Marshall W. Freeman*
PRESIDENT *Donald A Pazour*
SENIOR VICE PRESIDENT/CFO *Warren "Andy" Ambrose*
SENIOR VICE PRESIDENTS *H. Ted Bahr, David Nussbaum, Darrell Denny, Wini D. Ragus, Galen A. Poss*
VICE PRESIDENT/SOFTWARE DIVISION *Regina Starr Ridley*
VICE PRESIDENT/PRODUCTION *Andrew A. Mickus*
VICE PRESIDENT/CIRCULATION *Jerry Okabe*
GROUP PUBLISHER *Peter Hutchinson*

***DR. DOBB'S JOURNAL* ISSN 1044-789X** is published monthly by Miller Freeman, Inc., 600 Harrison Street, San Francisco, CA 94107; 415-905-2200. Periodicals Postage Paid at San Francisco and at additional mailing offices.

ARTICLE SUBMISSIONS: Send manuscript and disk (with article, listings, and letter to the editor) to *DDJ* Submissions, 411 Borel Ave., San Mateo, CA 94402-3522.

***DDJ* ON COMPUSERVE:** Type GO DDJ.

***DDJ* ON THE INTERNET:** editors@ddj.com or http://www.ddj.com/.

SUBSCRIPTION: $34.95 for 1 year; $69.90 for 2 years. International orders must be prepaid. Payment may be made via Mastercard, Visa, or American Express; or via U.S. funds drawn on a U.S. bank. Canada and Mexico: $45.00 per year. All other foreign: $70.00 per year.

FOR SUBSCRIPTION QUESTIONS, change of address, and orders, call toll-free **800-456-1215** (U.S. and Canada). For all other countries, call **303-678-0439** or fax **303-661-1885**. On CompuServe, the customer-service number is 71572,341. Or write, *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80322-6188.

BACK ISSUES may be purchased for $9.00 per copy (includes shipping and handling). For issue availability, call **800-444-4881** in the U.S. and Canada. All other countries call **785-838-7500** or fax **785-841-2624**. Send e-mail to **orders@mfi.com.** Back issue orders must be prepaid. Please send payment to *Dr. Dobb's Journal*, 1601 W. 23rd Street, Suite 200, Lawrence, KS 66046-2700.

POSTMASTER: Send address changes to *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80328-6188. **ISSN 1044-789X. GST (Canada) # R124771239.**

Canada Post International Publications Mail Product (Canadian Distribution) **Sales Agreement No. 0548677.**

FOREIGN NEWSSTAND DISTRIBUTOR: Worldwide Media Service Inc., 30 Montgomery St., Jersey City, NJ 07302; 201-332-7100.

Estimated print run 160,000.

Miller Freeman
A United News & Media publication

ABP American Business Press

Printed in the USA

BPA INTERNATIONAL

Is It Real Time, or Is It Microsoft?

More than once, Microsoft's marketing knuckles have been rapped for its nasty little habit of preannouncing products. Not that rebukes seem to matter, if Microsoft's recent preannouncement for adding "hard" real-time capabilities to Windows CE is any indication. According to a somewhat nebulous press release, Windows CE will be a "hard" real-time operating system with the release of Version 3.0 sometime in the second quarter of 1999. Interestingly, at last fall's Embedded Systems West Conference, Microsoft was careful not to call Windows CE 2.0 "hard" real time at all. Then suddenly, like pigs sprouting wings, Microsoft was referring to WinCE as a hard RTOS at this spring's Windows CE Developers Conference. (Soft real time is more forgiving than hard. Soft real time can miss deadlines in cases where not completing tasks is more acceptable than a failure. Hard real-time deadlines, on the other hand, must always be met. At minimum, hard real time must be deterministic, have low latency, and support nested INTs.)

When asked about this in a *DDJ* web site "Online OP-ED" interview (http://www.ddj.com/), a WinCE product manager cleared things up, explaining that the 1999 release of WinCE 3.0 will be "true" hard real time, implying that WinCE 2.0 is some other kind of hard real time. Yes, by the most minimal of definitions, WinCE 2.0 is a RTOS—but it's about as hard as butter on your morning biscuits.

WinCE has a long way to go before it can truly be called hard real time—especially when compared to tried and tested RTOSs such as QNX, VRTX, VxWorks, pSOS, and the like. (For instance, some WinCE latency figures are measured at from 93–275 microseconds; under QNX, comparable figures are at about two microseconds.) In all likelihood, a total rewrite of the WinCE kernel will be required to bring WinCE up to par with real RTOSs. But for all we know, of course, that rewrite is underway.

If history has taught us anything about Microsoft, it is that the company has a hard time meeting promises when it comes to shipping operating-system products—especially when those products are announced more than a year in advance. Having more resources than most of us can imagine didn't necessarily get Windows 95/98/NT 5.0 out the door when promised, making you wonder why Windows CE should be any different.

So why does Microsoft keep on preannouncing operating-system products so far in advance? More than likely to freeze the marketplace until a minimal implementation of what's promised can be delivered. Clearly, that marketplace would be better served by walking the walk, instead of talking the talk.

***

About the same time Microsoft was exhibiting chutzpah in the real-time realm, O'Reilly & Associates was down the road patting itself on the back over its self-proclaimed "historic" Open Source Summit. According to its press releases, O'Reilly brought together "heavyweights of the Internet software community…to explore ways of expanding the use and acceptance of open source software development."

No question, this is an admirable goal that deserves all of our support. The invitation-only event included the likes of Linus Torvalds, Larry Wall, Brian Behlendorf, John Ousterhout, Guido van Rossum, Phil Zimmermann, John Gilmore, Eric Raymond, Tom Paquin, Jamie Zawinski, Sameer Parekh, Eric Allman, Greg Olson, and Paul Vixie—each of whom deserves accolades for his contribution to the world of software development.

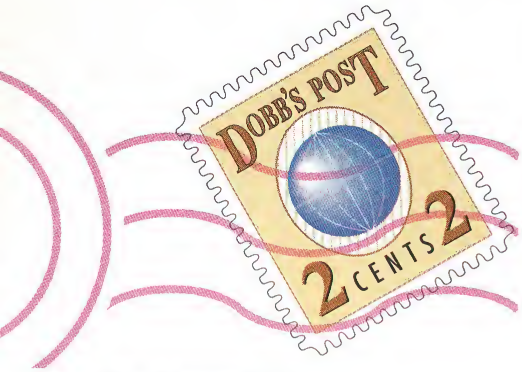
More noticeable, however, was who wasn't invited. If any single person deserves credit for launching the open source software movement, it's Richard Stallman of GNU and free software fame. An "open source summit" without Stallman is like a cheeseburger without the cheddar.

When, in response to a flurry of O'Reilly e-mail press releases, I asked by reply why Stallman wasn't invited, the net went suddenly quiet. Inquiring minds want to know.

***

A recent study by Software Success (http://www.softwaresuccess.com/) revealed a couple of interesting twists. The analysis, compiled by Software Success using data supplied by Dun & Bradstreet, showed that the total number of companies competing in the software industry grew from 58,779 in July 1997 to 68,765 in March 1998. For the first time since 1993 (when Software Success started tracking this data), the rate of growth of mid-sized companies was faster than that of startups. For instance, the number of companies with annual sales of under $500,000 (50,482) increased 12 percent since July 1997, the number with sales of $1 million–$5 million increased 42 percent, and those with $10M and up increased 83 percent. Software Success also found that the number of companies in the software-related services sector grew 45 percent to 19,542, reflecting a migration of some formerly product-based companies to the services sector.

Jonathan Erickson
Jonathan Erickson
editor-in-chief
jerickson@ddj.com



Y2K Challenges

Dear *DDJ*,

In the article "Date Compression and Year 2000 Challenges," by Robert L. Moore and D. Gregory Foley (*DDJ*, May 1998), I was surprised to see the Windowing concept referred to as a preferred method for meeting the much closer Year 2000 deadlines. Surprising, only in that I've previously suggested the same method, and your discussion of the approach shows I'm not totally out of touch with reality.

In Robert and Gregory's discussion, one aspect of Windowing may benefit from an alternate approach.

In dealing with sorting of Y2K-deficient data, they mentioned an interpretive program would need to be activated for converting non-Y2K data into a sortable Y2K-compliant format. A secondary interpretive program would then need to be written for converting the sorted Y2K-compliant output back into its native format. Instead of this approach, if achievable, the following would be preferred.

Have the OS *sort* utility modified to include a sort option that incorporates a Windowing interpretation. In the JCL, a one-byte field would indicate that sort requires the sort Windowing option. This field (code) would be available for each sort field. As with Windowing, the sort parameters would contain a singular field for specifying the Windowing (pivot) year. The sort would perform an on-the-fly Y2K interpretation while sorting the data, without actually modifying or expanding file contents.

I realize many OSes may be in use and are no longer upgradable. With the billions of dollars which will need to be spent on Y2K compliance, I would think enough clout could be established to coerce someone to make the required modifications.

An alternative, the interpretative programs (used before and after each sort) described in the article could be designed to utilize the JCL passed codes, similar to the sort parameters context in order to determine which fields need to be converted. By using variable parameters of this nature, it wouldn't be necessary to write a separate set of programs for each sort.

Storing the pivot year in the Working Storage Section was also recommended. If only one program needed modification for Y2K compliance, that would be great. However, thousands of programs are going to be modified. To adhere to a Sliding Windowing concept, storing the pivot year in Working Storage would require massive modifications just to affect a new Window. Perhaps a singular file containing the definition of the Windowing method/period could be created (for storage of any periodic variable data). Then each affected program would open the file, extract the windowing data, then store the data in Working Storage.

After this was integrated into all programs utilizing the Sliding Windowing concept, modifying the pivotal year file would simultaneously affect all programs.

Wayne H. Wilhelm
whw96sv@cardnet.stark.k12.oh.us

Dear *DDJ*,

One problem with all the compression schemes mentioned by Robert L. Moore and D. Gregory Foley in their article "Date Compression and Year 2000 Challenges" (*DDJ*, May 1998) is that human readability is lost. ASCII only makes use of the seven least significant bits of each word. Using the most significant bit from each of the six characters used to represent a date by the MMDDYY method, and using a base year of 1900, we can extend the present method to 64 centuries. Hopefully in that time we can work out a better system. Dates printed by a routine that strips off the most significant bit will still be human readable.

Lloyd C. Brown
Lloyd.Brown@gat.com

Dear *DDJ*,

I congratulate Robert L. Moore and D. Gregory Foley on their clear, well-written article "Date Compression and Year 2000 Challenges" (*DDJ*, May 1998) that focuses on the fundamental engineering problem of the Y2K "situation." At work, I have had to complete many spurious Y2K forms and questionnaires from customers who just don't get it, and who have latched on to the four-digit year as a mantra to protect themselves from Y2K ruin. With Robert and Gregory's article, perhaps I can teach them to converse rationally about the subject (one can always hope).

However, I was disappointed about a slight omission in the discussion—the issue of backward compatibility of storage. As mentioned in the article, there are two goals in programming a Y2K fix: to provide a representation for all dates the system could possibly need, and to do this with a minimum of programming effort (including software maintenance). The authors also mention that compression methods can reduce the amount of coding required to fix Y2K problems. You can reduce that effort even further if you don't have to convert all your persistent data to a new representation.

All of the compression methods provided in the article use the entire "value space" of each representation. As such, they all collide with the legacy representation. For example, the six characters "012001" could mean January 20, 1901 (MMDDYY) or January 1, 1912 (CYYDDD) or January 20, 12337 (MMDD 16b-year). So there is no way to examine a date to determine the encoding scheme used. Implementing these representations requires that all existing data be converted before the new software may be used, and that the old software is fully retired before the change.

Namespace techniques can be used to remove this burden, by designing the new representation to be complementary with the legacy representation. As mentioned in the article, the MMDDYY format makes very sparse use of the 48 bits required for storage; all of the methods described by the authors can be modified to exclude the normal MMDDYY representations from their "value space" and still retain sufficiently large ranges of dates. For instance, modify the CYYDDD format so that values of C start at "2"; values of "0" and "1" would indicate data in the old format. This still provides 900 years of dates, but allows the program to read data in both CYYDDD and MMDDYY formats. Similar tricks will work with each of the other formats described—I leave the details as an exercise to the reader.

By using a backward-compatible compression scheme, the need for updating existing data sources to the new representation is removed. To implement the fix, we only need to reprogram the data interface (read from/write to storage), possibly adjusting the internal date format and user output to account for the increased range of dates. Then release the revised program to users. In my experience, this is the minimum effort required to correct a Y2K deficiency.

Curtis S. Carney
awiggin@slip.net

Java and CORBA

Dear *DDJ*,

In "Building Distributed Applications with Java and CORBA" (*DDJ*, April 1998), Bryan Morgan does a good job with outlining to intricacies of CORBA. I do find, however, that I cannot agree with some of his statements and findings.

First, CORBA is not a vendor-independent operating system. The Object Management Group (OMG) never intended CORBA to re-

(continued from page 10)

place the core operating system of any node in an *n*-tier client/server environment. Bryan's statement can leave someone thinking that CORBA can replace NT or UNIX. Had Bryan stated that CORBA provides a vendor-independent environment for inter-object management across a network, I would have agreed with him.

Secondly, the proper use of the Internet Inter-ORB Protocol (IIOP) is somewhat of a religious war among CORBA proponents. Bryan flippantly discusses IIOP as a "wire-level protocol that resides on top of TCP/IP." He further states that IIOP "lets one vendor's CORBA 2.0-compliant ORB exchange objects with another's." Bryan is certainly stating the promise of IIOP rather than the fact. Let's look at the facts:

- Since CORBA is a suite of guidelines and does not dictate how a vendor should implement its CORBA-compliant solutions, inter-operability among the various ORBs is less than ideal—even with IIOP.
- IIOP is a good beginning toward addressing ORB inter-operability, but falls short of synchronizing such CORBA services as security and time across ORBs. Companies building CORBA-based applications are advised to choose a single ORB vendor and remain as homogeneous as possible. Mixing and matching ORBs is risky business in today's client/server world.
- One of the original intentions of IIOP was to build a bridge between CORBA and the Distributed Computing Environment (DCE). CORBA proponents recognize that while DCE is falling from favor in the *n*-tier client/server environment, it provides mature network services for building and managing client/server applications. DCE, unlike CORBA, is an industry standard rather than a suite of guidelines. As an industry standard, DCE limits the implementation variation across vendors. Moreover, some of the most robust security and authorization facilities have grown out of the DCE standard, like Kerberos.
- Many CORBA-compliant vendors currently have stable products for CORBA 1.0—fewer have stable products for CORBA 2.0. Since IIOP is part of the CORBA 2.0 specification, it would not make sound business sense to use IIOP to integrate ORBs from two or more vendors without knowing how each vendor has implemented their respective CORBA services (this defeats the purpose of encapsulation at the ORB service level, a foundation of object management).

CORBA is certainly the wave of the future. Since CORBA is an evolving specification, it is important that forethought and prudence are used to ensure that we build feature-rich and robust object-based client/server applications. As Java replaces C++ as the developer's tool of choice for building client/server applications, we must create greater awareness of what is real and doable, versus what is promised.

Richard S. Kravchuk
richard.kravchuk@ey.com

Window Sizes and the Registry

Dear *DDJ*,

Thanks to Al Stevens for the info in his April 1998 "C Programming" column on how to solve the problem with a window that could either be maximized or minimized. I had a clean install of Microsoft office on my machine. The only problem was that Microsoft Photo Editor refused to be anything but maximized or minimized. After reading Al's column, I searched the registry and found InitialPosition=65500,2,66112,565. I deleted that and all is well now. Not a big deal, kind of annoying, so I never went too far in finding out the problem.

Kevin Peck
KPeck@bridge.com

DDJ

ADDING IMAGING?

not without

LEADTOOLS

If you have an imaging project in front of you, choose the #1 toolkit in the market. LEADTOOLS is an award winning imaging toolkit that puts more than 7 years of development and millions of lines of code at your finger tips. You will save countless hours and have access to the best imaging technology available. Imaging technology that Microsoft, Hewlett Packard, Corel, Xerox and thousands of others have chosen for their products. With LEADTOOLS, your project is almost done!

VIEWING & SPECIAL EFFECTS

LEADTOOLS provides optimized rendering of any image to all display devices with monitor calibrations, auto-dithering, scaling, zooming, scrolling, and animation. Choose from 113 Paint Effects, 64 Dissolve Effects, and 24 Transition Effects, including pushes, pulls, wipes, splits, blinds, crushes, rolls, circulars, diagonals, stretches and many more with delays, grain sizes, pattern brushes (up to 64 passes), a colored wand and a transparent color. The combinations are unlimited. Other Special Effects include: 3D Shapes, 3D Rotated Text, 3D Frames, Gradient and Pattern Filled shapes and much more.

INTERNET/INTRANET

LEADTOOLS features a Net Aware ActiveX and a Netscape plug-in for Internet/Intranet applications, including a Bitmap Datapath allowing images to be read from any URL, Progressive JPEG, Progressive CMP, and support for GIF interlace, transparency, animation, and embedded text. A FeedLoad function has been created to allow image data to be displayed as it is being transmitted across the net.

DATABASE

LEADTOOLS has specific features designed for the imaging database developer: VB data binding, 32-bit ODBC, a customized OLE 2.0 in-place server, Load/Save memory, and Load/Save file offset.

LEADTOOLS is backed up by a 30-day money back guarntee (US & Canada only) and FREE technical support is available via phone, fax, Internet, CompuServe or BBS.

SCANNING

LEADTOOLS supports high speed scanning using ISIS and TWAIN. TWAIN includes both 16 & 32 bit native and buffered RAM transfer modes.

PRINTING

LEADTOOLS performs all image processing necessary to print directly to any Windows supported printer, with the ability to print text and multiple images on the same page.

COMPRESSION

LEADTOOLS offers more compression options than any other toolkit on the market, in both standard and proprietary formats.

THE POWER OF LEADTOOLS

LEADTOOLS is a collection of more than 400 functions, properties and methods that provide low level functions for complete control, all the way to the highest level functions for ease of use. Imaging Common Dialog Boxes, (new to v. 9.0) greatly simplifies integration. Royalty free and available as 16 & 32 bit DLLs, 16 & 32 bit ActiveXs, and a VBX and includes extensive source code examples for Microsoft Visual C/C++, Borland C/C++, Microsoft Visual Basic, Borland Delphi, Microsoft Visual FoxPro, Microsoft Access, VB and Java Script.

NOW FEATURING EASY TO USE
LEADTOOLS IMAGING COMMON DIALOGS

"LEADTOOLS is an indispensable tool, we used it in the development of our FrontPage application".

Tom Button
Director of Marketing,
Internet Platform & Tools division
Microsoft

"The file format support is phenomenal, LEADTOOLS gave Micrografx's applications (like CreataCard™ and Picture Publisher™) the flexibility to import and export a wide variety of established and new file formats with ease. Their format implementations are complete and performance Great!"

Andy Cohen
Director Software Development
Consumer Products
Micrografx

90

MORE THAN 50+ FILE FORMATS

JPEG	WMF
DICOM	WPG
DIB	AVI
EPS	WAV
AVI	MIDI
ICO	MSP
CUR	VDA
DCX	GIF
IMG	-Embedded Text
IOCA	-Animated
MODCA	-Interlaced
FAX[raw]	-Transparency
WINFAX	TIFF 6.0
CALS raster	-MPT
GEM	-JPEG
PNG	-Packbits
PSD	-Huffman
MAC	-RLE
Photo CD (PCD)	-LZW
FlashPix (FPX)	-CMYK
Exif	-CCITT
TGA	-CCITT G31D
RAS	-CCITT G32D
PCT	-CCITT G4
CMP	-Bitonal
BMP	-Grayscale
OS2 BMP	-RGB
PCX	-YCbCr
AWD	-CIELAB

IMAGE PROCESSING

TRANSFORMS

Including resize, resample (interpolated resize), rotate (.01 degree), flip, invert, reverse, crop, underlay, shear, transpose, fill, auto deskew and combine bitmap (with mathematical and Boolean operations).

FILTERS

Including sharpen, blur, brighten, darken, invert, hue and saturation, intensity, contrast, gamma correction, histogram equalize, edge detect, line detect, emboss, mosaic, posterize, median and noise filters, spatial filter (which can be predefined such as gradient, laplacian, sobel, prewitt, shift and difference, line segment, or they can be customized), and more.

DRAWING

Draw directly to a bitmap surface using any of the windows GDI functions (such as TextOut, BitBlt, Ellipse, and Rectangle).

REGION OF INTEREST

Process only a specific portion of an image rather than the entire bitmap. Regions can be comprised of any combination of rectangles, ellipses, rounded-rectangles, freehand shapes, polygons, transparent color and more.

CUSTOMIZE NOW

FlashPix™ MODULE

Integrate the full FlashPix experience.
- FlashPix viewing transforms.
- FlashPix non-image data.
- FlashPix thumbnails.
- Load and Save portions of an image.

CUSTOMIZE NOW

VIDEO MODULE

Integrate extended multimedia capabilities.
- Motion video editing.
- Record, Create, Edit, Play and Save AVI files.
- Record, Play, and save WAV files.
- Play Midi files.

ALSO AVAILABLE

PRO EXPRESS

Top of the line LEADTOOLS toolkit.
- Document Imaging.
- Annotation.
- Xerox TextBridge® OCR.
- Bi-tonal Filters.

Order now: 800-637-1837
900 Baxter St. Charlotte, NC 28204
704-332-5532 Fax: 704-372-8161 CompuServe:"GO LEADTECH"

LEAD TECHNOLOGIES INCORPORATED

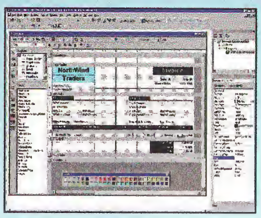
Special Offers, Evaluations, Demos & more! http://www.leadtools.com/

Programmer's Paradise®

the developer's *definitive* source for software!

The Hottest Development Tools! Order Now!

Paradise Pick of the Month!



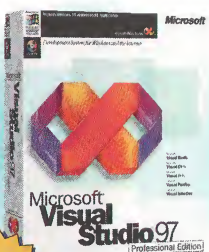
ActiveReports
by Data Dynamics

For VB developers who need to generate a wide variety of reports from their applications, ActiveReports is an ActiveX Designer that is powerful and easy to learn. Unlike any other report writer, our product takes full advantage of VB's powerful, familiar language and data binding capabilities.

NEW!

$275

Paradise No. D03 0210-FT



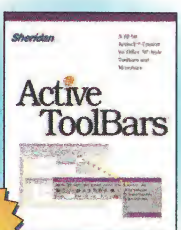
$100 Rebate*

$329.95*

Paradise No. M47 0112-FT

Visual Studio 97 Pro
by Microsoft Corp.

** Price after $100 manufacturer's rebate.*



New Product

$145

Paradise No. S16 0810-FT

ActiveToolBars
by Sheridan Software Systems

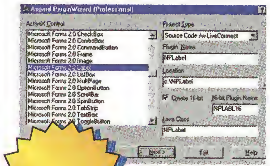
WISE Installation System Enterprise Edition
by WISE Solutions

The WISE Installation System Enterprise Edition incorporates the award-winning WISE Installation System, and the tools required for any type of application deployment. The Enterprise Edition includes SmartPatch, WebDeploy, and SetupCapture. The WISE Installation System creates professional installation programs for Windows, Windows 95, and Windows NT. WISE is a completely Windows-based installation editing/testing environment.

Paradise No. G10 0230-FT

$645

PluginWizard™
by Asgard



Harness the power of ActiveX controls in Netscape Navigator. Asgard PluginWizard will quickly generate Netscape Plugins from your own or 3rd party ActiveX controls. The standard edition directly creates a non-scripting plugin executable, while the professional edition generates full LiveConnect plugin source code. Royalty free!

NEW!

Paradise No. A66 0110-FT

$235

Visual SlickEdit
by MicroEdge

This award-winning editor increases development productivity, reduces costs of software maintenance and improves software quality through powerful features, software standardization and compatibility with your existing environment.

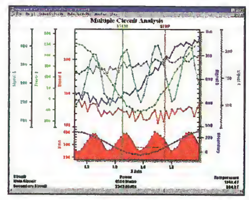
Software Standardization

With its multi-platform presence, integration with industry leading development environments and compatibility with version control systems, Visual SlickEdit provides your entire organization with a standard coding environment.

Paradise No. M39 0122-FT

$285

ProEssentials™
by GigaSoft®



16- and 32-bit DLL, OCX, VCL, and VBX interfaces providing charting functionality with consistent visual quality, real-world practicality, and overall professional appeal. Suited for engineering, financial, data-acquisition, and information system development. Be sure to find the best tool for your important development needs. Download a **demo** [500K] or get a **fully functional evaluation edition** [3Meg] at **www.pparadise.com/publishers/gigasoft**.

Paradise No. G04 0110-FT

$345

ClassMagic
by Object Dynamics

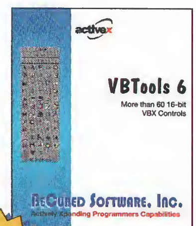
Assemble Windows applications, COM components or DLL in C++, using your current tools, libraries and code! Cut in half the code you write—create modular, thread-safe classes, connect objects, assemble new classes from interconnected objects, package them together. Includes the ClassMagic engine, library of reusable software parts, source samples, complete documentation and much more.

New!

Paradise No. 014 0110-FT

$479

VBTools 6
by BeCubed Software



Update your 16-bit applications today with any of the more than 60 custom 16-bit VBX controls! You'll save bunches of time with the new International control because it assigns internationalized text to all loaded controls! And there's a Flow Charting control, a 2D Slider, a Floating Text Extender and more, plus many other enhancements!

New Version!

Paradise No. B30 0310-FT

$95

Multi-Edit 8
by American Cybernetics

Multi-Edit 8 is the only 32-bit source-code editor a programmer needs. Fully customizable, it ships with extensive multi-language support including Web languages, IDE and VCS Integration and syntax highlighting. Run compilers and other tools in the background. Cut keystrokes in half with language-specific Smart Indenting, Template Editing, and Construct Matching.

Paradise No. A30 0110-FT

$149

Spread v2.5
by FarPoint Technologies

A complete spreadsheet control for most environments that support a VBX, OLE control, or DLL. Use as a spreadsheet to obtain variable lines of data or display tables of information. Data-aware—connect to databases with the Access Engine and ODBC. Includes over 250 properties and our improved Spread Designer. Formulas, sorting, and full-print support too.

Paradise No. F02 0110-FT

$225

Order online 24 hours a day, every day at www.pparadise.com

Visit **www.pparadise.com** *for the full story on these and more great products! Use our product search button & order online 24 hours a day, every day!*

The Fastest & Easiest Way to Create Help Systems

by Blue Sky® Software

RoboHELP 6.0
The Best Selling Help Authoring Tool*

RoboHELP is the fastest and easiest way to create online Help for any platform. RoboHELP turns Microsoft Word into a full-featured authoring tool capable of creating professional Windows Help, WebHelp, Windows 98 Help (Microsoft HTML Help), Netscape NetHelp, printed documentation, intranet/Internet Web sites, and Windows CE Help—all from a single source.

** Based on independent market survey.*

RoboHELP Office 6.0
The Complete Help Authoring Solution

RoboHELP Office provides a complete Help authoring solution with a suite of powerful tools for authoring nine professional Help formats, including the new Windows 98 Help format. Includes the award-winning RoboHELP, RoboHTML (at no extra cost–$499 value), plus over 16 powerful Help authoring utilities.

RoboHELP Office
Paradise No.
B13 0240-FT
$659**

*** Price after $100 manufacturer's mail-in rebate.*

RoboHELP
Paradise No.
B13 0140-FT
$469

Paradise Pick of the Month!

TestTrack

by Seapine Software

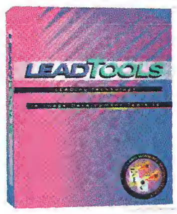
TestTrack Web — Simply Better Bug Tracking

TestTrack is the fastest and most complete multi-user bug tracking solution for Windows 95/NT. Tracks bug and feature requests, customers, users, test configurations, and more. Advanced features include e-mail notifications, e-mail bug import, duplicate bug handling, release note generation, and much more. Distribute TestTrack's standalone bug reporter to your customers to automate customer support. With all of its power, TestTrack remains the easiest bug tracking solution to use and maintain.

TestTrack
Paradise No. S96 0210-FT
$469

LEADTOOLS ActiveX Pro 32

by LEAD Technologies, Inc.



LEADTOOLS offers technology in all major imaging categories for color, grayscale and bitonal imaging, with comprehensive functionality in each category. LEADTOOLS is an integrated development toolkit with more than 500 functions, properties and methods. Common Dialog boxes make LEADTOOLS easier to use and add-on modules are available for additional FlashPix, OCR, and Video support.

Paradise No.
L05 0132-FT
$335

RoboHTML

by Blue Sky® Software

Easily Create Windows 98 Help

RoboHTML is the only product specifically designed to create the new Windows 98 HTML Help standard. RoboHTML provides a rich WYSIWYG editor with full drag and drop support, automated project management, and complete testing features. Unlike HTML editors, RoboHTML has built-in support for all Windows 98 specific features such as: dynamic Table of Contents; multi-level Indexes; and Related Topics. Also supports easy creation of Popups, Navigation Buttons, Splash Screens, Shortcuts, Import existing WinHelp projects and more.

Paradise No.
B13 0320-FT
$469

A Personal Guarantee!

"If you're not happy, we're not happy! You have my personal guarantee that we'll provide you with outstanding products, support and service."

—Roger Paradis
President & CEO
Programmer's Paradise, Inc.

Here's just a small sampling of what you'll find in the Programmer's Paradise catalog. To find all the tools you need to succeed call us today!

Doc-To-Help

by Wextech Systems

Create HTML Help, Windows Help for 95, NT 3.51/4.0 and 3.0/3.1, HTML, and printed documentation from one file, at the touch of a button, with Doc-To-Help 3. Wextech, the Microsoft acknowledged expert in HTML Help authoring, incorporates our experience in Doc-To-Help 3, enabling you to create great HTML Help.

Paradise No.
W08 0120-FT
$399

KEDIT for Windows 1.5

by Mansfield Software Group

KEDIT for Windows

Includes both 32-bit Windows 95/Windows NT and 16-bit Windows 3.1 modules. KEDIT is a powerful general purpose text editor with redefineable keys, undo/redo, selective line editing, regular expression support, enhanced syntax coloring, column oriented editing, file locking, a macro debugger, an IBM XEDIT-compatible command set, prefix area support, and more! The macro language is a subset of REXX. Also available in DOS and OS/2 text mode versions.

Paradise No.
MAN KE31-FT
$135

ORDER NOW

or call for your FREE catalog!

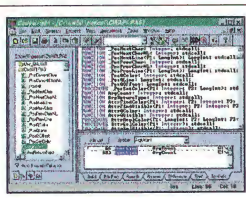
1-800-445-7899

Call for shipping charges/return policy.
Prices subject to change without notice.

NASDAQ®: PROG

Codewright Pro

by Premia



You'll work faster and easier when you use the right programmer's editor, Codewright. Now you'll browse code faster with Outline Symbols. It gathers information about your code in the background. You'll juggle changes with ease, with Difference Editing. It lets you selectively combine the changes from two revisions. The new Bookmarks Window lets you view bookmarks by name and by file. Now with synchronizing technology for Delphi and Visual C++ IDEs. With the help of the API Assistant, making complex function calls is as simple as filling in a form.

Paradise No.
P36 0111-FT
$225

List Pro

by FarPoint Technologies

A set of customizable List Box and Combo Box controls. Create header groups and sub header groups and/or multiple columns to create a very unique list control. Merge cells in rows with same text to make display more user friendly. Bind to supported databases using Microsoft Jet Database Engine or Microsoft Access ODBC. Virtual memory manager allows connection to any size database. Supports single or multi-select lists, embedded icons and bitmaps in columns and rows, plus drag-and-drop capabilities.

Paradise No.
F02 0211-FT
$185

To Order Call 800-445-7899 or FAX 732-389-9227

Dr. Dobb's News & Views

And the Winner Is...

The West Coast trounced the East Coast for the seventh time in ten tries at The Computer Museum's Tenth Annual Computer Bowl, a trivia contest held in Boston (West Coast fans gathered in Silicon Valley at Moffett Field's Hangar One to participate via video simulcast) that pits teams of five industry luminaries against each other. The West Coast, garbed in cowboy hats and led by Netscape cofounder Marc Andreesen, dominated the event, beating the frilly-shirt-wearing East Coast, 230–70. John Ratzenberger (aka Cliff Clavin, mailman and trivia buff from the TV show *Cheers*) was master of ceremonies of the event, asking questions ranging from the highly technical ("How many times more bandwidth does a T1 line have than a 56 kilobaud modem?") to the highly obscure ("How many microprocessors are there on Mars?"). Pride and fine food were at stake, which raises money for the museum—Sunnyvale, California, Mayor James Roberts won a lobster feast from a bet with Boston, Massachusetts, Mayor Thomas Menino. For more information, see http://www.computerbowl.org/.

Biometric Security Moves Forward

SAC Technologies (http://www.sacman.com/), a biometrics security company that provides technology for network and computer security without the use of pin numbers, passwords, or tokens, has received certification from the International Computer Security Association (http://www.icsa.net/). Certification was in the one-to-many Identification category. Identification is the process of comparing the biometric characteristics of an unknown individual against characteristics stored in a database to determine their identity. Identification asks, "Who is this?" and establishes whether more than one biometric record exists, thus denying an individual who is attempting to pass himself off with more than one identity.

Don't Blink

A "PIN-less" automatic teller machine (ATM) has gone online at the Nationwide Building Society bank in Great Britain. The system, designed by NCR, uses a biometric iris-identification system developed by Sensar (http://www.sensar.com/). To use the system, bank customers simply insert their ATM card into a reader and a camera mounted in the machine compares the customer's iris (one of the few human body parts to remain unchanged as aging occurs) with records in the databank. The process takes as little two seconds.

Sensar uses iris-recognition software developed by IriScan (http://www.iriscan.com/). The software is also being tested in Virginia by Spring Technologies as an automated fare-collection system in mass-transit applications. The goal of this automated system, called "TranScan," is to expedite commuter entry and exit at subway and train stations by minimizing and eventually eliminating the commuter's need to insert a card, pass, or token.

Macro Writing Contest

Premia Corp. has announced a macro writing contest for Premia's Codewright Programmer's Editor. The contest is being run in conjunction with the addition of Perl, AppBasic, and API (C-like) macros in Codewright 5.1. The grand prize for the best macro is $5000, or one of a number of other prizes. In addition, there will be first, second, and third place prizes for macros written in each of the three macro languages. Submissions must be received no later than August 1, 1998. Winners will be announced at the SD '98 East Conference in Washington, D.C. on August 18, 1998. For more information, see http://www.premia.com/.

E-Stamps on the Way

The U.S. Post Office has approved electronic postage stamps (e-stamps) for testing and, if things go as expected, we'll be printing our own stamps using PCs and the Internet. E-stamps include the postage amount, name and zip code of the local post office, date the postage was printed, and rate category (first class or whatever). In addition, e-stamps will have electronic bar coding of the same information as well as the identification number of the printing device and a digital pattern that will make each envelope unique and hard to counterfeit.

The system approved for testing, called "SmartStamp," was developed by E-Stamp Corp. (http://www.e-stamp.com/). Other approaches, such as PostagePlus from Neopost (http://www.neopost.com/), are coming online too. SmartStamp requires dongle-like hardware that fits into a printer port, serving as an electronic vault for postage. PostagePlus, on the other hand, requires no additional hardware. Customers will have an account with e-stamp companies and can download postage into this vault via the Internet.

Déjà Cygnus

Over the last few years, Metrowerks' CodeWarrior (http://www.metrowerks.com/) development tools have been extended from their Mac origins to include support for a wide array of languages (C, C++, Object Pascal, and Java), processors (including x86, PowerPC, MIPS, and Java VM), and systems (BeOS, PowerStation, Windows, and so on).

One of CodeWarrior's biggest competitors is GNU GCC, which has good support for cross-compilation to a variety of processors. To better appeal to companies that have standardized on GNU GCC, Metrowerks now officially supports the GNU GCC compiler from within the CodeWarrior environment (as an alternative to Metrowerks' own compiler). A new subsidiary, Quorum Technologies, has been formed for the express purpose of supporting GCC within CodeWarrior.

Cryptographers Crack Cell-Phone Code

Taking only about six hours of work, cryptographers at the University of California at Berkeley cracked Global System for Mobile Communications (GSM) codes, enabling them to "clone" a digital cellphone and make unauthorized calls from another phone. In the process, Ian Goldberg, David Wagner, and Marc Briceno also discovered indications that the code may have been intentionally weakened during its design. The GSM digital standard is the most widely used in the world, with more than 79 million phones in use.

Worldwide PC Sales Climb

According to a recent report by market-research firm Dataquest, sales of personal computers continue to grow at double-digit rates. Overall, says Dataquest, worldwide PC shipments were up 14.1 percent for the first quarter of 1998, compared with the same period in 1997. U.S. growth was 16.2 percent.

As for who's leading the vendor pack, Compaq maintained its market-share lead with 12.5 percent worldwide and 17.1 percent in the U.S. Dell Computer weighed in with an 11.7 percent in the U.S. Worldwide shipments by Hewlett-Packard and Dell were up 72 percent and 66.1 percent, respectively, over the last year.

Composing Reactive Animations

Programming for greater freedom of expression

Conal Elliott

There's no question that computer graphics—especially interactive graphics—is an incredibly expressive medium with potential beyond imagination. However, few people are able to create interactive graphics, so what might be a widely shared medium of communication is instead a tool for specialists. The problem is that authors still have to worry about how to get a computer to present content, rather than focus on the nature of the content itself. For instance, behaviors such as motion and growth are generally gradual, continuous phenomena; moreover, many such behaviors go on simultaneously. Computers cannot directly accommodate either of these basic properties, because they do their work in discrete steps rather than continuously, and they only do one thing at a time. Graphics programmers consequently have to bridge the gap between *what* an animation is and *how* to present it on a computer.

If the kind of programming in use today (like that described in the accompanying text box "Models versus Presentations" on page 25) is unsuitable for most potential authors, then we need to move toward a different form of programming. Alternative forms must give authors freedom of expression to say what an animation is, while invisibly handling details of discrete, sequential presentation. In other words, these forms must be declarative ("what to be"), rather than imperative ("how to do").

Conal is a member of the Microsoft Research Graphics Group. He can be contacted at conal@microsoft.com.

In this article, I present one such approach to declarative programming of interactive content. Fran (short for "functional reactive animation") is a high-level vocabulary that lets you describe the essential nature of an animated model, while omitting details of presentation. And because this vocabulary is embedded in a modern functional programming language (Haskell), the animation models are reusable and composable in powerful ways.

Fran is freely available (with source code) as part of the Hugs implementation of Haskell for Windows 95/NT (http://www.haskell.org/hugs/). Newer versions of Fran may be found at http://www.research.microsoft.com/~conal/Fran/. The underlying ideas form the basis of Microsoft's DirectAnimation, a COM-based programming interface accessible through conventional languages like Java, Visual Basic, JavaScript, VBScript, and C++. DirectAnimation is built into Internet Explorer 4.0, so you may already have it.

There are three ways you can experience this article:

- In this printed version, examples have an accompanying sequence of snapshots. By scanning them from left to right, top to bottom (first row, second row, and so on), you'll get a sense of motion.
- On the Web (http://www.research.microsoft.com/~conal/Fran/tutorial.htm), examples are illustrated by animated GIFs, showing animation over time, but not interactivity. That version of this article also contains additional discussion and several animations not in the printed version.
- Finally, you can run the examples and interact with or modify them. After installing Hugs (available at http://www.haskell.org/hugs/), double-click on the file tutorial.hs in the subdirectory lib\Fran\demos. At the > prompt, type "main" and press Enter. The examples will begin running. Press Spacebar, "n," or right arrow to advance to the next animation, and "p" or left arrow for the previous one. If you want to display just a single animation (*leftRightCharlotte*, for instance), then close the animation window and enter

"display leftRightCharlotte". You can alter the definition in an editor, save the result, enter ":r" to the Hugs prompt, and "$$" again to display the new version. For 2D examples having a user argument *u*, use *displayU* instead of *display*. Similarly, for 3D examples, use *displayG* if there is no user argument, and *displayGU* if there is a user argument.

The First Example

I'll start with the animation in Figure 1 called *leftRightCharlotte*, which moves Charlotte from side to side. Listing One (listings begin on page 20) defines a value called *leftRightCharlotte* to be the result of applying *moveXY* to three arguments. (In most other programming languages, you would instead say something like "moveXY(wiggle,0,charlotte)".)

The function *moveXY* takes *x* and *y* values and an image, and produces an image moved horizontally by *x* and vertically by *y*. All values may be animated. In this example, the *x* value is given by *wiggle*, a predefined smoothly animated number. *Wiggle* starts out at 0, increases to 1, decreases back past 0 to –1, and then increases to 0 again — all in the course of two seconds, and then it repeats, forever. The second line defines *charlotte* by importing a bitmap file, making it available for use on the first line as the second argument to *moveXY*.

Although this example isn't a masterpiece, it is nonetheless a complete animation program in just two short lines of code.

Similarly, Figure 2 and Listing Two define an animation of Patrick moving up and down. To get the vertical movement, I've used a nonzero value for the second argument to *moveXY*. Rather than using *wiggle*, you use *waggle*, which is defined to be just like *wiggle*, but delayed by half a second.

Figure 3 and Listing Three combine the two previous examples. The *over* operation glues two animations together, yielding a single animation, with the first one being over the second. Because I used *waggle* for *upDownPat* in this combined animation, Pat is at the center when Charlotte is at her extremes (and vice versa).

Composition

Composition is the principle of putting together simple things to make complex ones, then putting these together to make even more complex things, and so on. This building-block principle is crucial for making even moderately complicated constructions; without it, the complexity quickly becomes unmanageable.

Listings One through Three illustrate composition. I first built *leftRightCharlotte* out of *charlotte*, *wiggle*, and *moveXY*; then *upDownPat* out of *pat*, *moveXY*, and *waggle*. Finally, I built *charlottePatDance* out of *leftRightCharlotte* and *upDownPat*. A crucial point here is that when you make something out of building blocks, the result is a new building block in itself, and you can forget about how it was constructed.

There is a more powerful version of composition, based on defining functions. Listing Four, for instance, defines *hvDance* (for "horizontally and vertical dance"), which combines any two images, in the way that *charlottePatDance* combines *charlotte* and *pat*. Now you can give a new definition for the dancing couple that gives exactly the same animation: *charlottePatDance* = *hvDance charlotte pat*.

Having defined this generalized dance animation, you can go on to more exotic compositions. For example, you can take an animation produced by *hvDance*, shrink it, and put the result back into *hvDance* twice to make it dance with itself. As Figure 4 and Listing Five show, the result is pleasantly surprising. This example gives you a hint of how powerful it is to be able to define new animation functions. For instance, you could try *charlottePatDance*, stretched by a wiggly amount; see Listing Six(a). To prevent negative scaling, you take the absolute value of *wiggle*. Next, use *hvDance* again, but give it wiggly sized *charlotte* and *pat*. For visual balance, use *wiggle* and *waggle*; see Listing Six(b). Next, put Pat in orbit around a growing and shrinking Charlotte. To get a circular motion, use *moveXY*, with *wiggle* for *x* and *waggle* for *y*; see Listing Six(c).

As you may have surmised, *wiggle* and *waggle* are related to *sine* and *cosine* and defined as:

```
waggle = cos (pi * time)
wiggle = sin (pi * time)
```

The animated number *time* is a commonly used "seed" for animations and has the value *t* at time *t*. Thus, for instance, the value of *wiggle* at time *t* is equal to $sin(\pi t)$.

Rate-Based Animation

Up to now, the positions of animations have been specified directly. For instance, the definition of *leftRightCharlotte* says that Charlotte's horizontal position is *wiggle*.

In the physical universe, objects move as a consequence of forces. As Newton explained, force leads to acceleration, acceleration to velocity, and velocity to position. With computer animation, you have the freedom to ignore the laws of our universe. However, since animations are usually intended to be viewed by and interacted with by inhabitants of our own universe, they are often made to look and feel real by emulating Newtonian laws or simplifications and variations on them.

The key idea underlying Newton's laws and their variations is the notion of an instantaneous rate of change. Fran makes this notion available in animation programs. To illustrate rate-based animation, you can make Becky move from the left edge of the viewing window, toward the right, at a rate of one distance unit per second; see Figure 5 and Listing Seven.

The local definition of *x* here (introduced as a *where* clause), follows a style you'll see in the following definitions. To express an animated value that starts out with a value *x0* and grows at

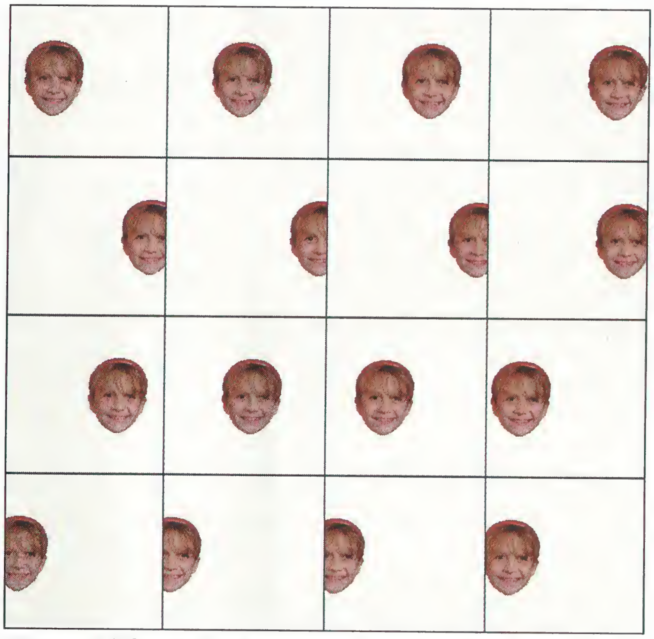


Figure 1: leftRightCharlotte *moves Charlotte from side to side.*

Listing One

```
leftRightCharlotte = moveXY wiggle 0 charlotte
charlotte = importBitmap "../Media/charlotte.bmp"
```

Listing Two

```
upDownPat = moveXY 0 waggle pat
pat = importBitmap "../Media/pat.bmp"
```

Listing Three

```
charlottePatDance =
  leftRightCharlotte `over` upDownPat
```

Listing Four

```
hvDance im1 im2 =
    moveXY wiggle 0 im1  `over`
    moveXY 0 waggle im2
```

Listing Five

```
charlottePatDoubleDance = hvDance aSmall aSmall
  where
    aSmall = stretch 0.5 charlottePatDance
```

Listing Six

```
(a)
dance1 = stretch (abs wiggle) charlottePatDance

(b)
dance2 = hvDance (stretch wiggle charlotte)
                 (stretch waggle pat)

(c)
patOrbitsCharlotte =
   stretch wiggle charlotte  `over`
   moveXY wiggle waggle pat
```

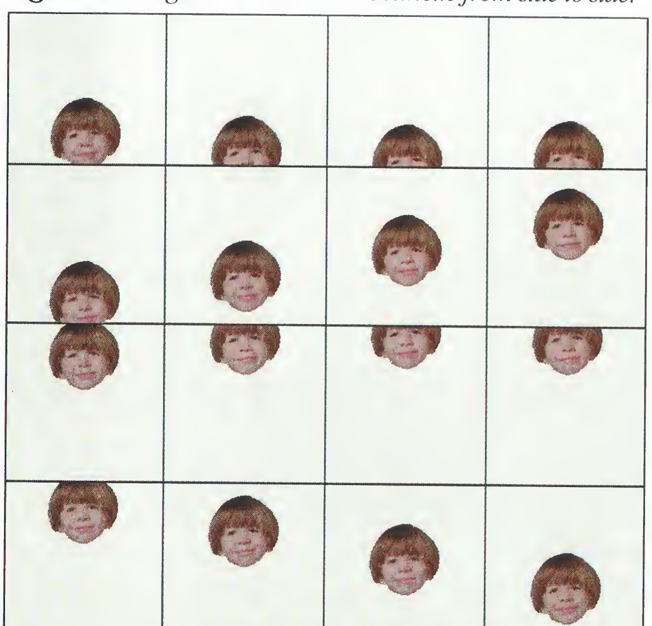


Figure 2: *Patrick moving up and down.*

Figure 3: *Combining Charlotte and Patrick.*

(continued from page 20)

a rate of *r*, you say *x0 + atRate r u*. Here *u* is a "user", which is a Fran value that contains all user input and display update events. Rate-based animations require a user argument in order to give *atRate* a way of knowing when to start and how precisely to calculate value from rate. Unlike previous examples, this one can be displayed with *displayU*. To see this example, enter *displayU velBecky*.

In Listing Seven, Becky has a constant velocity, but with a little more effort you can give Becky a constant acceleration by providing a constant value for the rate of change of the velocity; see Listing Eight. In the definition of *v*, the "0 +" is unnecessary, but emphasizes that the initial velocity is zero.

The notion of "rate" is useful not just in one dimension, but in two and three dimensions as well. In Listing Nine, I control Becky's 2D velocity with the mouse. When you hold the mouse cursor at the center of the view window, Becky stays still. As you move away from the center, imagine an arrow from the window's center to the mouse cursor. Becky moves in that direction and her speed will be equal to the arrow's length. This kind of imaginary arrow is referred to as a "vector" and is the same type of quantity as a two- or three-dimensional offset, velocity, or acceleration. In 2D, a vector can be thought of as having horizontal and vertical (X and Y) components, or as having a magnitude (length) and direction. This time, I use *move*, a variant of *moveXY* that takes a 2D offset vector. (If a vector *v* is *x* units horizontally and *y* units vertically, then "move v im" is equivalent to "moveXY x y im.") The offset vector starts out as the zero vector, and grows at a rate equal to *mouseMotion*, which is the offset of the mouse cursor relative to the origin of 2D space (which you see in the center of the view window).

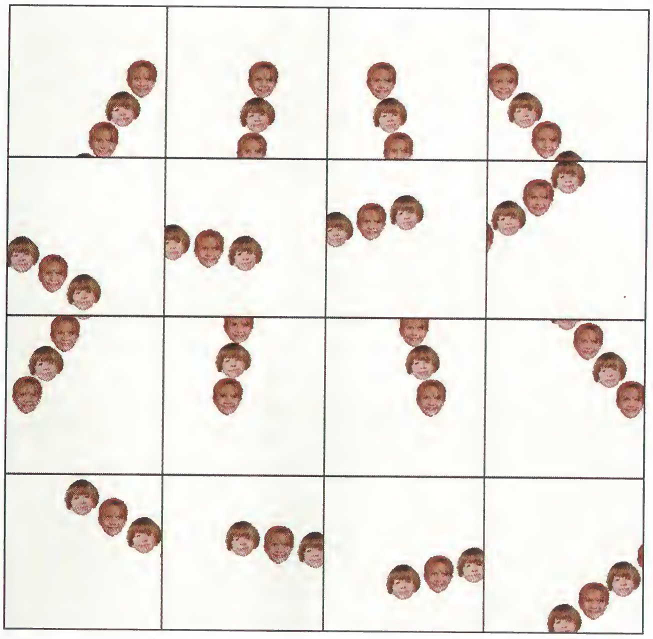


Figure 4: *Defining new animation functions.*

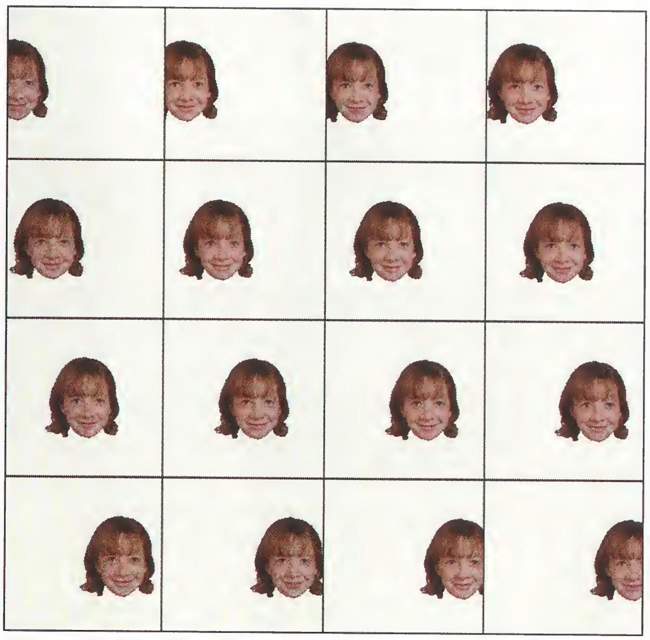


Figure 5: *Rate-based animation at a rate of one distance unit per second.*

In the real world, the position of an object may affect its speed or acceleration. In Listing Ten, Becky is chasing the mouse cursor. The further away it is, the faster she moves. The only difference from Listing Nine is that the velocity is determined by where the mouse cursor is relative to Becky's own position, as indicated by the vector subtraction.

For fun, you can generalize the *beckyChaseMouse* function in the same way that *hvDance* generalized *charlottePatDance* earlier; see Listing Eleven. Then *chaseMouse becky* is equivalent to *beckyChaseMouse*, as you can verify by typing *displayU (chaseMouse becky)* at the Hugs prompt.

For more fun, try the same, but replace *becky* with some of the animations that appeared earlier (*leftRightCharlotte*, *charlottePatDance*, and *patOrbitsCharlotte*); see Figure 6 and Listing Twelve.

Next make a chasing animation that acts like it is attached to the mouse cursor by a spring. The definition is similar to *beckyChaseMouse*. In Listing Thirteen, however, the rate is itself changing at rate *accel* (acceleration). This acceleration is defined like the velocity was in the previous example, but this time, some

Listing Seven

```
velBecky u = moveXY x 0 becky
  where
    x = -1 + atRate 1 u
```

Listing Eight

```
accelBecky u = moveXY x 0 becky
   where
     x = -1 + atRate v u
     v =  0 + atRate 1 u
```

Listing Nine

```
mouseVelBecky u = move offset becky
   where
    offset = atRate vel u
    vel    = mouseMotion u
```

Listing Ten

```
beckyChaseMouse u = move offset becky
   where
    offset = atRate vel u
    vel    = mouseMotion u - offset
```

Listing Eleven

```
chaseMouse im u = move offset im
   where
    offset = atRate vel u
    vel    = mouseMotion u - offset
```

Listing Twelve

```
danceChase u =
   chaseMouse (stretch 0.5 charlottePatDance) u
```

Listing Thirteen

```
springDragBecky u = move offset becky
   where
    offset = atRate vel u
    vel    = atRate accel u
    accel  = (mouseMotion u - offset) - 0.5 *^ vel
```

(continued from page 22)

drag is also added. This tends to slow down Becky by adding some acceleration in the direction opposite to her movement. (Increasing or decreasing the "drag factor" of 0.5 in Listing Thirteen creates more or less drag.) The operator *^ multiplies a number by a vector, yielding a new vector that has the same direction as the given one but a scaled magnitude.

As usual, these declarative animation programs are straightforward because they say what the motion is, in high-level, continuous terms, without struggling to accommodate the discreteness of the computer used to present them. In contrast, imperative animation programs must explicitly simulate rate-based animation by making lots of discrete steps — accumulating approximations

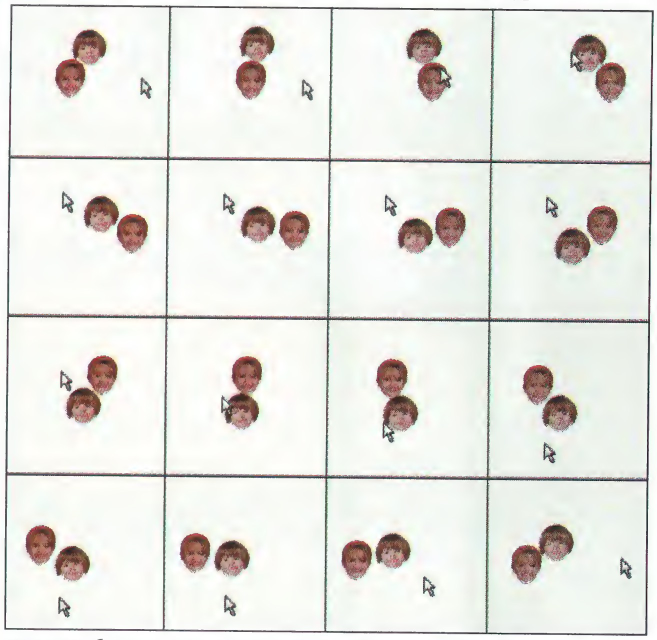


Figure 6: *Rate-based animation, but replacing Becky with animations such as* leftRightCharlotte, charlottePatDance, *and* patOrbitsCharlotte.

Figure 7: *Composition-in-time. Defining an orbiting animation, and then combining it with a version of itself delayed by one second.*

to the continuously varying forces, accelerations, and velocities—to approximate motion. Doing an accurate and efficient job of all this approximation work is a tricky task. With systems like Fran, you just describe the continuous motion in terms of continuously varying rates, and trust Fran to do a good job with the approximation. (Not good enough to fly an airplane or control dangerous machinery, but good enough for an effective illustration or game.)

Composition-in-Time

Operations such as *over* and *move* support the principle of composition-in-space. Composition-in-time is equally valuable. Figure 7 and Listing Fourteen, for instance, define an orbiting animation, and then combine it with a version of itself delayed by one second. Instead of delaying, you can speed it up; see Listing Fifteen. You can even delay or slow down animations involving user input. In Listing Sixteen, one Jake tracks the mouse cursor, while the other follows the same path, but delayed by one second.

Next you can build an animated sentence, following the mouse's motion path. As a preliminary step, use *delayAnims dt anims = overs (zipWith later [0, dt ..] anims)* to define a *delayAnims* function, which takes a time delay *dt* and a list *anims* of animations, and yields an animation. Each successive member of the given animation list is delayed by the given amount after the previous member. The definition of *delayAnims* introduces a few new Fran elements. The Fran *overs* function is like *over*, but applies to a list of animations rather than just two. Animations earlier in the list are placed over ones later in the list. The notation *[0, dt ...]* means the infinite list of numbers 0, *dt*, 2 *dt*, 3 *dt*, and so on. Finally, *zipWith* applies to a given two-argument function the successive values from two given lists. You use it here to delay the first animation in *anims* by 0 seconds, the second by *dt* seconds, the third by 2*dt* seconds, and so on. Finally, *overs* combines them into a single animation. Figure 8 and Listing Seventeen present a simple use of *delayAnims*. Next, use *delayAnims* (Listing Eighteen) to define *mouseTrailWords* that makes animated sentences.

The Haskell *words* function takes a string apart into a list of separate words. The Haskell *map* function takes a function (*moveWord*) and a list of values (the separated words) and makes a

Models versus Presentations

Here is a rough sketch of the steps you usually go through to program an animation:

```
Allocate and initialize window, various drawing surfaces and bitmaps
  repeat until quit:
        get time ( t)
        clear back buffer
        for each sprite (back to front):
        compute position, scale, etc. at t
        draw to back buffer
        fast copy ("blit") back buffer to front
  Flip back buffer to the screen
Deallocate bitmaps, drawing surfaces, window
```

These steps are usually carried out with lots of tedious, low-level code you have to write yourself. Most of this work is not about what the animation is, but how to present it. In contrast, Fran programs are only about what the animation is.

—C.E.

Listing Fourteen

```
orbitAndLater = orbit `over` later 1 orbit
  where
    orbit = moveXY wiggle waggle jake
```

Listing Fifteen

```
orbitAndFaster = orbit `over` faster 2 orbit
   where
    orbit = move wiggle waggle jake
```

Listing Sixteen

```
followMouseAndDelay u =
  follow `over` later 1 follow
   where
    follow = move (mouseMotion u) jake
```

Listing Seventeen

```
kids u =
   delayAnims 0.5
     (map (move (mouseMotion u))
          [jake, becky, charlotte, pat])
```

Listing Eighteen

```
trailWords motion str =
  delayAnims 1 (map moveWord (words str))
  where
     moveWord word = move motion (
                     stretch 2 (
                       withColor blue (stringIm word) ))
```

Listing Nineteen

```
flows u = trailWords motion
             "Time flows like a river"
   where
     motion = 0.7 *^ vector2XY (cos time)
                               (sin (2 * time))
```

Listing Twenty

```
flows2 u = trailWords (mouseMotion u)
             "Time flows like a river"
```

Listing Twenty-One

```
redBlue u = buttonMonitor u `over`
            withColor c circle
  where
    c = red `untilB` lbp u -=> blue
```

Listing Twenty-Two

```
redBlueCycle u = buttonMonitor u `over`
                 withColor (cycle red blue u)
                           circle
  where
   cycle c1 c2 u =
    c1 `untilB` nextUser_ lbp u ==> cycle c2 c1
```

Listing Twenty-Three

```
tricycle u =
    buttonMonitor u `over`
    withColor (cycle3 green yellow red u) (
      stretch (wiggleRange 0.5 1)
        circle )
  where
   cycle3 c1 c2 c3 u =
    c1 `untilB` nextUser_ lbp u ==>
    cycle3 c2 c3 c1
```

Listing Twenty-Four

```
jumpFlower u = buttonMonitor u `over`
               moveXY (bSign u) 0 flower
flower = stretch 0.4
           (importBitmap "../Media/rose medium.bmp")
bSign u = selectLeftRight 0 (-1) 1 u
```

Listing Twenty-Five

```
growFlower u = buttonMonitor u `over`
               stretch (grow u) flower
grow u = size
  where
   size = 1 + atRate rate u
   rate = bSign u
```

Listing Twenty-Six

```
growFlowerExp u = buttonMonitor u `over`
                  stretch (grow' u) flower
grow' u = size
  where
   size = 1 + atRate rate u
   rate = bSign u * size
```

new list by applying the function to each member of the list. The Fran *stringIm* function makes a picture of a string. I define the function *moveWord* locally to be the result of making a picture of the given word, using the Fran *stringIm* function, and moving it to follow the mouse. *delayAnims* then causes each of these mouse-following word pictures to be delayed by different amounts. Figure 9 and Listing Nineteen is a use of *trailWords* following a specified path, while Listing Twenty follows the mouse.

Reactive Animation

The animations presented to this point can be called "nonreactive" since they always do the same thing. A "reactive" animation, on the other hand, involves discrete changes due to events. To illustrate, you can make a circle that starts off red and changes to blue when the left mouse button is pressed.

An informal reading of the last line of Listing Twenty-One (also see Figure 10) is that the color *c* is red until you press the left mouse button, then becomes blue. For a more literal reading, you must understand that there are really two new binary infix operators here—*untilB* and -=>—which can be used separately or together. Implied parentheses are around *lbp u* -=> *blue*. The -=> operator, which can be read as "handled by value," takes an event (*lbp u*) and a value (*blue*), and yields a new event. In this case, the new event happens when the left button is pressed, and has value *blue*. The *untilB* operator takes an animation of any type (the color-valued constant animation *red*), and an event (*lbp u* -=> *blue*), whose occurrence provides a new animation of the same type.

Cyclic Reactivity

To make Figure 10 more interesting, you can switch between red and blue every time the left button is pressed. As Listing Twenty-Two shows, you do this with the help of a *cycle* function that takes two colors (*c1* and *c2*) and gives an animated color that starts out as *c1*. When the button is pressed, it swaps *c1* and *c2* and repeats (using recursion).

Listing Twenty-Two uses the operator ==>, which is a variant of -=>. This operator (which can be read as "handled with function") takes an event and function *f*. It works like -=>, but gets event values by applying *f* to event values from the event given to it. In this case, *f* is the *cycle* function applied to just two arguments, leaving the third (a user) to be filled in automatically (using ==>). The *nextUser_* function turns *lbp* into an event

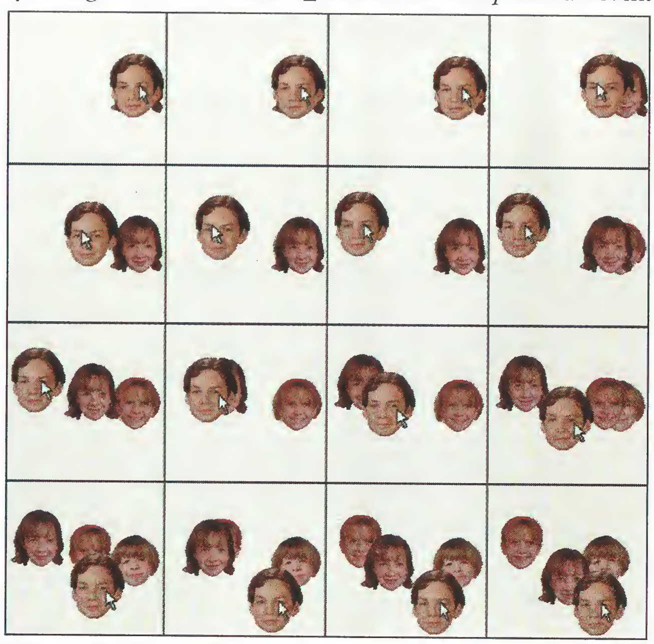
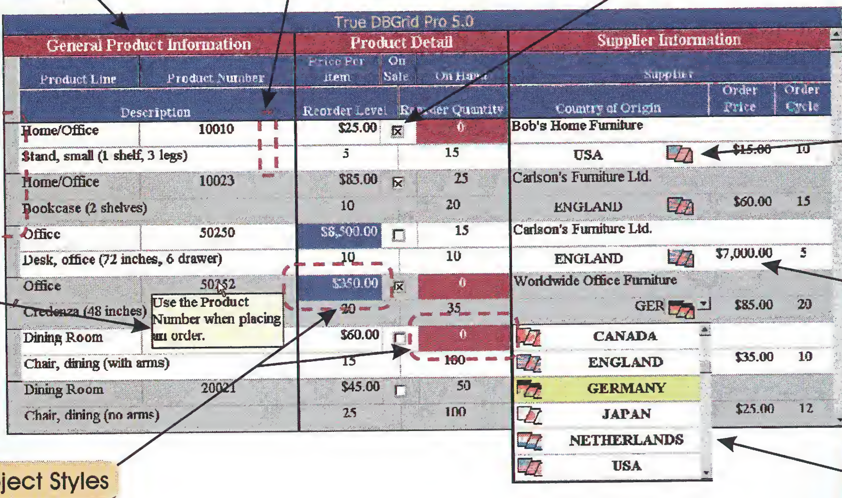


Figure 8: *Composition-in-time using* delayAnims.

Tried and true.



"The Official Upgrade Grid for Microsoft Visual C++"

Powerful, Easy-to-Use OCX Controls for VC++

You don't need to be an MFC expert to develop database applications with **APEX**'s award-winning components! Just plug an **APEX** OCX control into your Visual C++ project and see why developers in more than 400 of the Fortune 500 companies rave about **APEX**'s time-saving controls! **APEX**'s VC++ products have:

- ✓ Robust, feature-rich controls with the power and versatility to handle the most demanding database application requirements.
- ✓ Superior data handling technology to retrieve and display data faster and more efficiently than other third-party controls.
- ✓ Easy-to-use controls with VC++/VB cross-platform support and object models similar to those found in the Microsoft family of products.
- ✓ The best documentation in the industry, with extensive, well-organized online help and reference manuals, and numerous tutorials and samples.
- ✓ The best technical support team in the industry, dedicated to providing responses that are timely, accurate, helpful, and friendly.

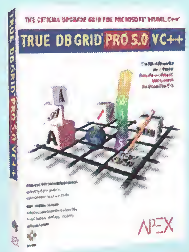
Download **FREE** demos and trial versions today!

TRUE DBGRID® PRO 5.0 VC++

True DBGrid Pro 5.0 VC++ is the Official Upgrade Grid to APEX's DBGrid, which is included in Microsoft Visual C++. Based upon the same data binding and user interface technologies as DBGrid, **True DBGrid Pro 5.0 VC++** adds over a hundred time-saving data presentation and editing features that give you complete control over the grid's appearance and run-time behavior. The fastest and easiest-to-use grid for VC++ also includes:

- ✓ Ability to access data using the MFC DAO Classes, RDO, ADO, ODBC, arrays, or your own custom data source
- ✓ **True DBGrid Pro 5.0 VC++** Object Class Library, which simplifies access to the grid's objects and collections
- ✓ 500 page manual and extensive online help, written specifically for VC++ developers
- ✓ One year of FREE technical support and product upgrades
- ✓ 10 tutorials and 10 sample projects
- ✓ and much more...

$399.95



*call for upgrade pricing

APEX®

(voice) 800.858.2739
(fax) 412.681.4343
(e-mail) sales@apexsc.com

APEX Software Corporation download FREE trial versions www.apexsc.com

(continued from page 26)
whose occurrence information is a new user, corresponding to the remainder of the user *u*. The color arguments get swapped each time "around the loop."

For variety, Listing Twenty-Three uses three colors, and changes the circle's size smoothly.

Selection

Figure 11 and Listing Twenty-Four present a flower that starts out in the center and moves to the left or right when the left or right mouse button is pressed, returning to the center when the button is released.

The function *bSign* is defined to be –1 when the left button is down, +1 when the right button is down, and 0 otherwise (thanks to *selectLeftRight*). You can use *bSign* to control the rate of growth of an image. In Figure 12 and Listing Twenty-Five, pressing the left (or right) button causes the image to shrink (or grow) until released. Put another way, the rate of growth is 0, –1, or 1, according to *bSign*. A simple change to the *grow* function (Listing Twenty-Six) causes the image to grow or shrink at a rate equal to its own size. *selectLeftRight*, used to define *bSign*, is also the key ingredient in defining *buttonMonitor* (Listing Twenty-Seven), which gives button feedback.

stringBIm turns an animated string into an image animation, which here gets enlarged, colored white, and moved down by a little less than half the window height.

selectLeftRight can itself be defined in terms of more basic functions, as in Listing Twenty-Eight. You use the conditional function *condB* to say that if the left button is down, use the left value, or if the right button is down, use the *none* value; otherwise use the *none* (*constantB*, which turns constants—nonanimations—into animations that never change).

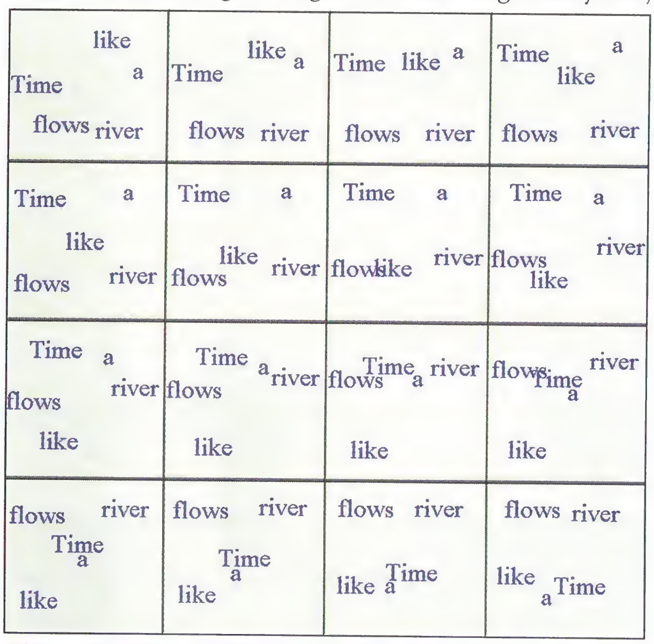


Figure 9: *Using* trailWords *following a specified path.*

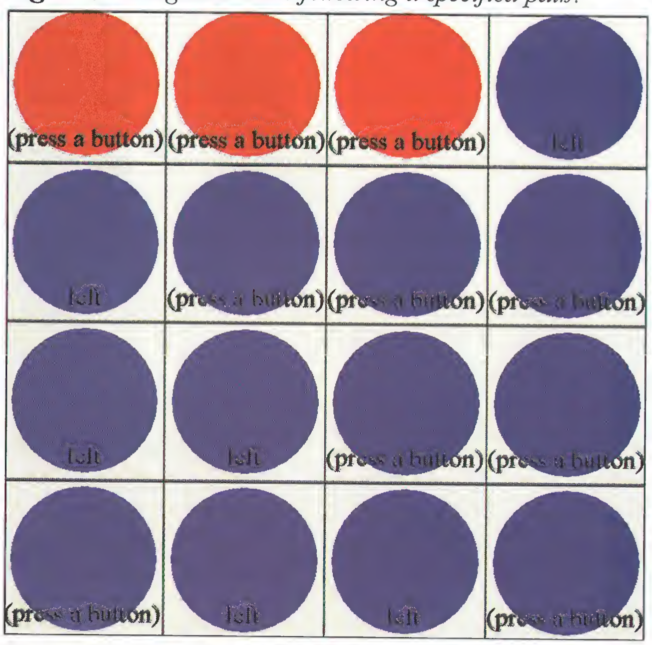


Figure 10: *The color* c *is red until you press the left mouse button, then it becomes blue.*

Listing Twenty-Seven

```
buttonMonitor u =
   moveXY 0 (- height / 2 + 0.25) (
    withColor textColor (
     stretch 2 (
     stringBIm (selectLeftRight "(press a button)" "left" "right" u))))
  where
    (width,height) = vector2XYCoords (viewSize u)
```

Listing Twenty-Eight

```
selectLeftRight none left right u =
   condB (leftButton u) (constantB left) (
    condB (rightButton u) (constantB right) (
      constantB none ))
```

Listing Twenty-Nine

```
teapot =
   stretch3 2 (importX "../Media/tpot2.x")
```

Listing Thirty

```
redSpinningPot =
  turn3 zVector3 time (
    withColorG red teapot)
```

Listing Thirty-One

```
mouseTurn g u =
   turn3 xVector3 y (
    turn3 zVector3 (-x) g)
  where
    (x,y) = vector2XYCoords (pi *^ mouseMotion u)
 mouseSpinningPot u =
   mouseTurn (withColorG green teapot) u
```

Listing Thirty-Two

```
spinPot potColor potAngle =
  turn3 zVector3 potAngle (
    withColorG potColor teapot)
```

Listing Thirty-Three

```
spin1 u = buttonMonitor u `over`
           renderGeometry (spinPot red (grow u))
                          defaultCamera
```

Listing Thirty-Four

```
withSpinner f u =
   buttonMonitor u `over`
   renderGeometry (f (grow u) u)
                  defaultCamera
```

Listing Thirty-Five

```
spin1 = withSpinner spinner1
  where
    spinner1 angle u = spinPot red angle
```

e-business™

To get a **free demo CD**, and to find out more about Electronic Commerce visit us at www.software.ibm.com/eci

Are all your applications working together toward a common goal? Your CEO wants to compete on a whole new playing field: Electronic Commerce. The fastest way to get there is by connecting your existing applications to each other and to the Web. Enter MQSeries® from IBM. It allows you to get your ordering, shipping, inventory and accounting systems to work together as a back-end to your e-commerce Web servers, without writing a lot of connectivity code. MQSeries lets you pipe messages between applications, while it takes care of program interfaces and supports Java™, COBOL, C++, Lotus Notes® and SAP.® It's available on 25 platforms, including Windows NT®, HP-UX, Sun™ Solaris® and all IBM platforms. MQSeries helps guarantee message delivery, even when programs or networks fail. So if you're new to the e-commerce game or you've been there for a while, MQSeries can put your business on a new playing field. Find out about getting your apps working like a team at www.software.ibm.com/eci

IBM®

Solutions for a small planet™

(continued from page 28)

3D Animation

Declarative animation applies to 3D as well, and the 2D operations I've used to this point—*importBMP*, *moveXY*, and *stretch*—have 3D counterparts. As a first 3D example, *sphere = importX "../Media/sphere2.x"* defines a sphere in which the function *importX* brings in a 3D model in "X-file" format, as used by Microsoft's DirectX. It is just as easy to import a teapot; see Figure 13 and Listing Twenty-Nine. I used *stretch3* (a 3D counterpart to *stretch*) because the imported model was too small. Listing Thirty colors the teapot and makes it spin around the z- (vertical) axis.

Next, you can use the mouse to control the teapot's orientation. To do this, define *mouseTurn* to turn a given geometry *g* around the x-axis according the mouse's vertical movement, and around the z-axis according the mouse's horizontal movement, scaled by π Finally, as Figure 14 and Listing Thirty-One show, you apply *mouseTurn* to a green teapot.

You can also make teapots spin by controlling the rotation angle with the *grow* function, as in the growing flower examples. First, define *spinPot*, see Listing Thirty-Two, that takes (animated) color and angle and yields a colored, turning teapot. Then make a pot that spins one way when the left button is pressed, and the other way when the right button is pressed, using the *grow* function, and giving feedback with *buttonMonitor*; see Figure 15 and Listing Thirty-Three. *renderGeometry*, used here with a convenient default camera, turns a 3D animation into a 2D animation.

Additional spinning teapots will all have the general form of using the button monitor and rendering with the default camera. Rather than having to write several definitions, give the pattern a name. In Listing Thirty-Four, *withSpinner* takes a function as its first argument, and applies that function to the result

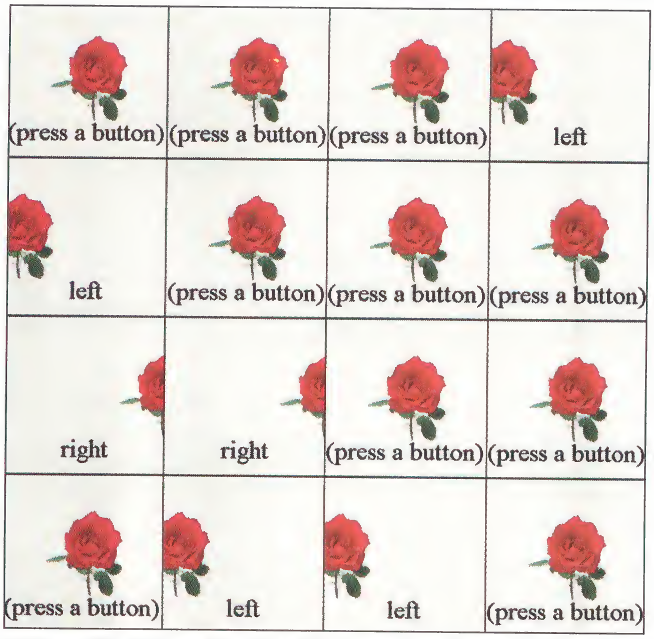


Figure 11: *Flower starts in the center and moves to the left or right when the left or right mouse button is pressed, returning to the center when the button is released.*

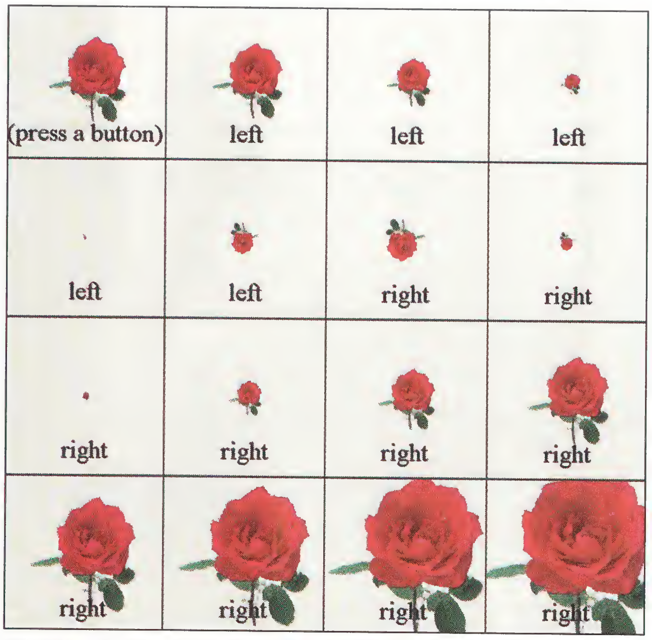


Figure 12: *Pressing the left (or right) button causes the image to shrink (or grow) until released.*

Listing Thirty-Six

```
spin2 = withSpinner spinner2
  where
    spinner2 potAngleSpeed u =
      spinPot (colorHSL time Ø.5 Ø.5)
              (atRate potAngleSpeed u)
```

Listing Thirty-Seven

```
sphereLowRes = importX "../Media/sphereØ.x"
movingLight =
   move3 motion (
    stretch3 Ø.1 (
     withColorG white (
      sphereLowRes `unionG` pointLightG)))
 where
  motion = vector3Spherical 1.5
             (pi*time) (2*pi*time)
potAndLight =
  withColorG green teapot `unionG` movingLight
```

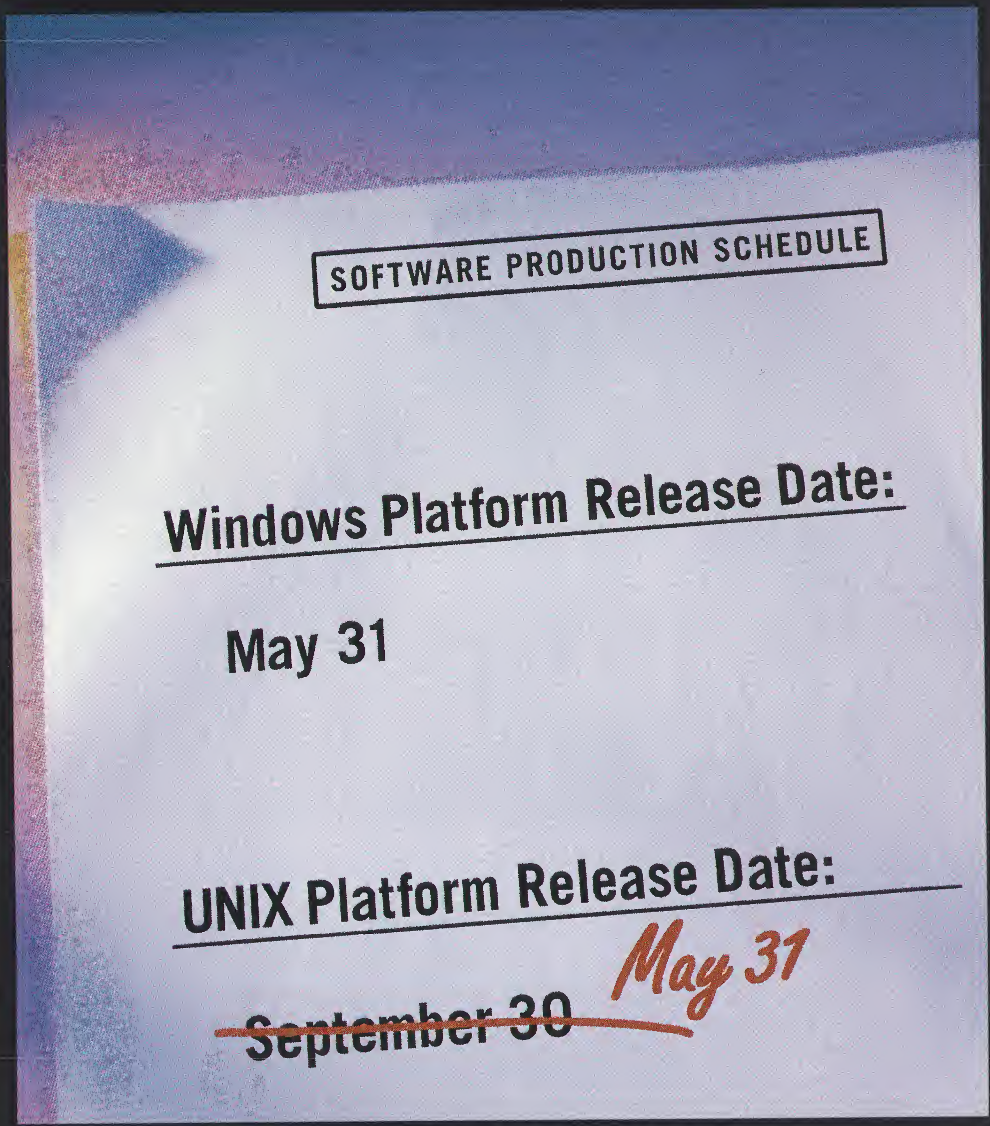
Listing Thirty-Eight

```
delayAnims3 dt anims =
   unionGs (zipWith later [Ø, dt ..] anims)
```

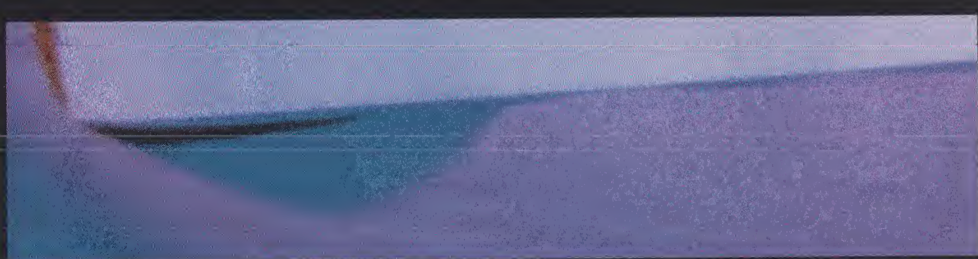
Listing Thirty-Nine

```
potAndLights =
   slower 5 (
    withColorG green teapot `unionG`
    delayAnims3 (2/5) (replicate 5 movingLight) )
```

Figure 13: *Importing a teapot.*



Use MainWin To Port NT Code To UNIX, And You'll Only Have To Rewrite One Line.



Fast. Flawless. Flexible. When you use the MainWin cross-platform toolkit, you write apps for Windows once, then port the source code to UNIX, using UNIX-native WIN32 APIs. There's no need to double your programming efforts, no need to rewrite your code.

WWW.MAINSOFT.COM

Plus, you can still retain the same Test and QA staff, and even use the same documentation for both versions. The comprehensive MainWin environment ensures native performance to your existing UNIX customers, while speeding your own migration to the expanding world of Windows NT.

MAINSOFT

You will significantly reduce your development costs and simultaneously launch new applications on Windows and UNIX. For an evaluation copy of MainWin XDE, call us, send email to democopy@mainsoft.com, or visit our website. A few seconds with us might save you months of hard work.

1-800-MAINWIN

(continued from page 30)

of the *grow* function applied to the user argument. With this definition, you can write *spin1* more simply; see Listing Thirty-Five. Another use of *withSpinner* is to make the color vary in hue and use the value from *grow* to determine the time-varying speed of rotation, so that the mouse buttons cause the turning to accelerate and decelerate (see Listing Thirty-Six).

In addition to visible geometry, you can add lights to a 3D model. In Listing Thirty-Seven, you combine a white sphere, which is visible but does not emit light, and a point light source, which is invisible but emits light. You color the sphere/light pair white, shrink it, and give it motion. For convenience, you express the motion path in terms of spherical coordinates, saying that the distance from the origin of space (which is also the center of the teapot) is always 1.5 units, the longitude is π times the elapsed time, and the latitude is twice π times the elapsed time. Consequently, you get a motion that meanders about, but maintains a fixed distance from the center of the teapot.

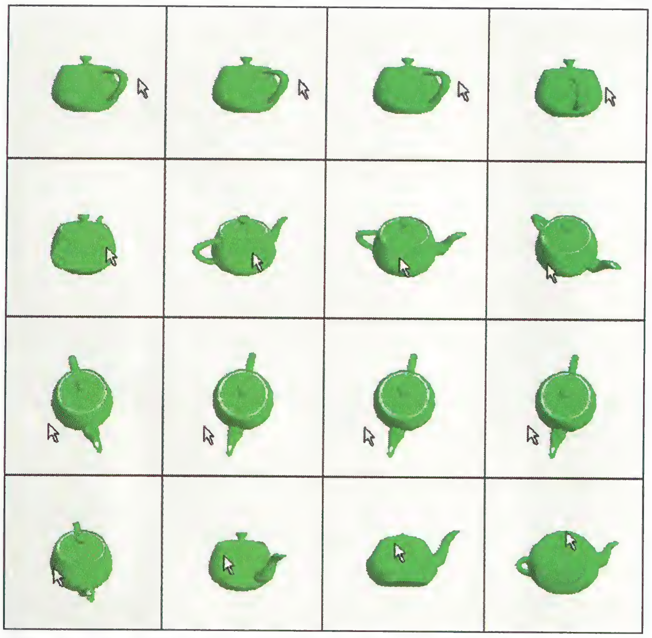


***Figure 14:** Applying* mouseTurn *to a green teapot.*

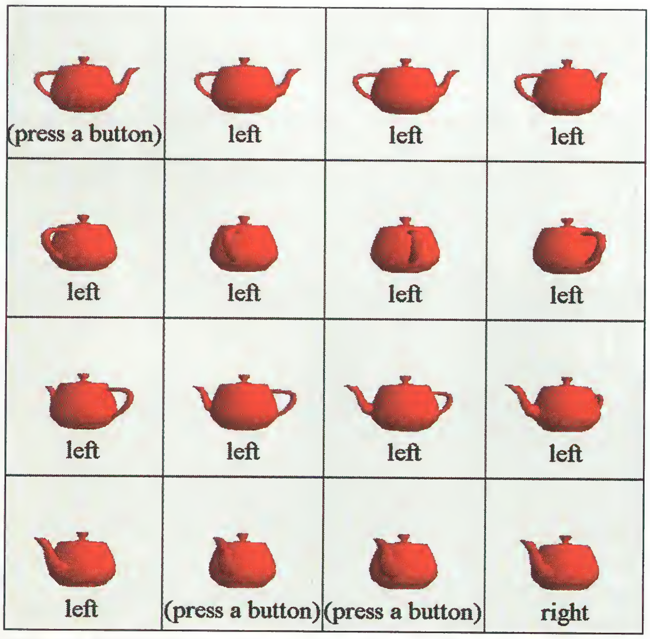


***Figure 15:** Making teapots spin by controlling the rotation angle with the* grow *function.*

Just for fun, replace the single moving light with five. A simple change suffices, if you add *delayAnims3*— a 3D variant of the 2D *delayAnims*. As Listing Thirty-Eight shows, the difference is that in the 3D version, you use *unionGs* instead of *overs*. With this function, you make a list of five copies of the moving light (see Listing Thirty-Nine), using the predefined Haskell function *replicate*, stagger them in time with *delayAnims3*, and combine them with a green teapot. Then slow down the animation to see it more clearly.

In Listing Forty and Figure 16 (a moving trail of colored balls), you define a single ball having a spiral motion, which traces the surface of an unseen sphere of radius 1.5 with a longitude angle changing ten times as fast as the latitude angle (five versus one-half radians per second). From this one moving ball, you make ten balls, each a differently colored version, and then stagger them in time with *delayAnims3*. The coloring function *bColor* produces evenly spaced hues.

As a final 3D example, Listing Forty-One presents another spiral. This time you form a static spiral, then turn it about the z-axis.

Listing Forty

```
spiral3D = delayAnims3 0.075 balls
  where
    ball   = move3 motion (stretch3 0.1 sphereLowRes)
    balls  = [ withColorG (bColor i) ball
             | i <- [1 .. n] ]
    motion = vector3Spherical 1.5 (10*time) time
    n      = 20
    bColor i =
      colorHSL (2*pi * fromInt i / fromInt n) 0.5 0.5
```

Listing Forty-One

```
spiralTurn = turn3 zVector3 (pi*time) (unionGs (map ball [1 .. n]))
  where
    n = 40
    ball i  = withColorG color (
                move3 motion (
                  stretch3 0.1 sphereLowRes ))
     where
       motion = vector3Spherical 1.5 (10*phi) phi
       phi    = pi * fromInt i / fromInt n
       color  = colorHSL (2*phi) 0.5 0.5
```

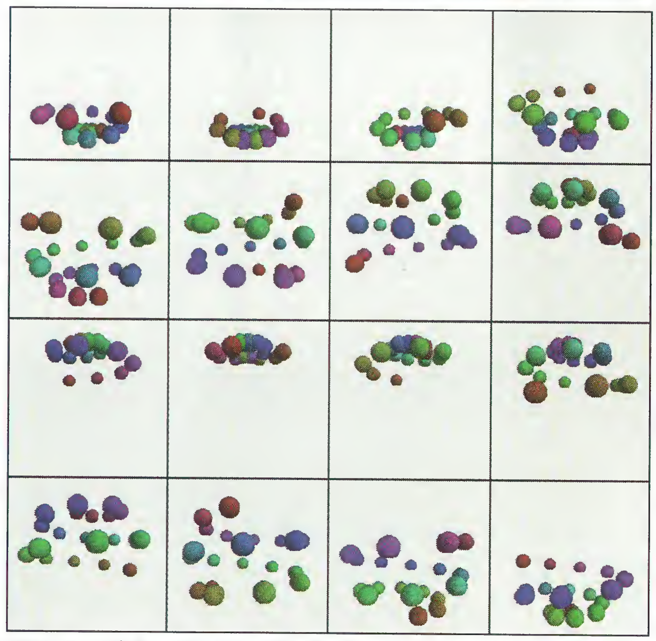


***Figure 16:** A moving trail of colored spheres.*

Related Work

My interest in functional animation originally started with Kavi Arya's "A Functional Approach to Animation," *Computer Graphics Forum*, 5(4):297-311 (December, 1986). Although elegant, Arya used a discrete model of time. The TBAG system, on the other hand, used a continuous time model, and had a syntactic flavor similar to Fran's; see "TBAG: A High Level Framework for Interactive, Animated 3D Graphics Applications," by Conal Elliott, Greg Schechter, Ricky Yeung, and Salim Abi-Ezzi (*Proceedings of SIGGRAPH '94* July, 1994). Unlike Fran, reactivity was handled imperatively. Behaviors were created by means of constraint solving, and updated through constraint assertion and retraction. Concurrent ML introduced a first-class notion of events that can be constructed compositionally; see "CML: A Higher-order Concurrent Language," by John H. Reppy (*Proceedings of the ACM SIGPLAN '91 Conference on Programming Language Design and Implementation*, 1991). However, those events perform side-effects such as writing to buffers or removing data from buffers. In contrast, Fran event occurrences have associated values—they help define what an animation is, but do not cause any side effects.

For examples of DirectAnimation, see http://www.microsoft.com/ie/ie40/demos and "Adding Theatrical Effects to Everyday Web Pages with DirectAnimation," by Salim AbiEzzi and Pablo Fernicola (*Microsoft Interactive Developer*, October 1997).

For background on Haskell, see *Introduction to Functional Programming*, by Richard Bird and Philip Wadler, (Prentice-Hall, 1987), "A Gentle Introduction to Haskell," by Paul Hudak and Joseph H. Fasel, *SIGPLAN Notices*, 27(5), May, 1992, and http://haskell.org/tutorial/index.html.

For information on Fran, refer to "Functional Reactive Animation," by Conal Elliott and Paul Hudak, *Proceedings of the 1997 ACM SIGPLAN International Conference on Functional Programming* (June, 1997), or the Fran web page at http://www.research.microsoft.com/mconal/Fran.

Conclusion

For interactive animation to expand into its potential as a medium of communication, it must become much easier to program. As this article illustrates, one step toward this goal is the replacement of imperative techniques ("how to do") with declarative ones ("what to be").

There are several features I haven't explored here, including sound, smooth flip-book animation, and cropping. There are also many opportunities for improvement: more features for 2D, sound, and 3D; improved efficiency; generation of animation "software components" to integrate with components written in more mainstream programming languages; and support for distributed, multiuser scenarios.

Acknowledgments

Todd Knoblock and Jim Kajiya helped to explore the basic ideas of behaviors and events. Sigbjorn Finne, Anthony Daniels, and Gary Shu Ling helped with the implementation during research internships. Alastair Reid made improvements to the Haskell code, and, along with Paul Hudak and John Peterson, provided helpful discussions about functional animation, how to use Haskell well, and lazy functional programming in general. Becky Elliott cut out the kid pictures, which appear with the kind permission of their owners Patrick, Charlotte, Becky, and Jake.

DDJ

A Conversation with John Knoll

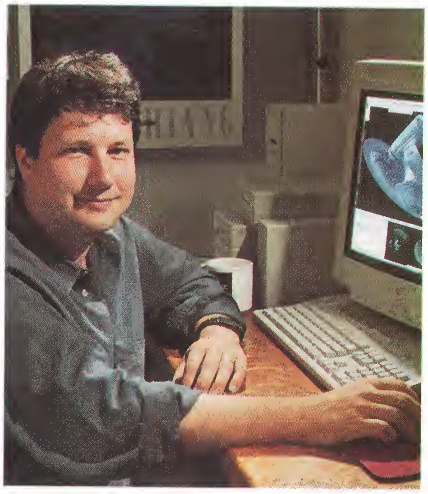
Life on the bleeding edge of computer graphics

Thomas "Rick" Tewell

As a visual effects supervisor for George Lucas' Industrial Light & Magic, John Knoll has lived on the bleeding-edge of computer graphics for over a decade. As such, he has worked on ground-breaking feature films such as *The Abyss* (which earned an Academy Award for Best Visual Effects), *Mission Impossible*, and *Star Trek VIII: First Contact*, among many others. He is currently working on the next *Star Wars* film, currently codenamed *Episode I*. In addition, John and his brother Tom are the creators of Adobe's PhotoShop image-processing software. John recently took time from his duties at Industrial Light & Magic in Marin County, California, to chat with Rick Tewell.

***DDJ*:** John, from what I understand, you transitioned from model-making into computer graphics. Can you tell us about that?
JK: Sure. When I was a kid, model-making was a hobby of mine. I got to be reasonably good at it and decided to go into visual effects as a career. I moved to Los Angeles to attend the University of Southern California film program. At USC, I tried to make contacts so that when I graduated, I wouldn't be going into an entry-level position. I was trying to get some of those entry-level-position years behind me while in school. So I started doing freelance model work.

***DDJ*:** Creatures or vehicles?
JK: Mostly the hard surface kinds of things. The first guy I worked for was Greg Jean who has a model shop. Since he runs a low-budget operation, he was happy to hire newbies and train us.



When the model was done, I'd take it out to the stage and fix things—during rigging, they'd need a hole here or something has got to move or I had to paint something to fix it because it didn't look good enough for camera. Somebody has to be around to do those sorts of things. So I would be on the stage a lot of the time when my models would be shot, which meant I got familiar with motion-control cameras. That was something that interested me. How do you get started doing that sort of thing? They didn't teach that at USC, which was mostly a live-action school. My last year at USC, I took an advanced animation class and we had a couple of manual hand-cranked animation stands. For my final project, I decided to build a simple four-channel motion-control system. This was in 1984. I bought a used Apple II and a four-channel serial-controlled CNC milling machine, which ran four stepper motors. And I bought a bunch of surplus stepper motors from C&H Sales and various bits and pieces. Although the camera got booked in two-hour blocks during the week, it was free during the weekend. Consequently, after the last session on Friday night, I could go in there, take the hand-cranks off, bolt my motors on, set up the computer, and shoot as long as I had it all cleared off by the first scheduled block on Monday. It was a lot of fun.

***DDJ*:** This was an Apple II?
JK: An Apple II Plus with a whopping 64K of RAM. I had a digital I/O board so I could control various relays.

***DDJ*:** So primarily, you were using the Apple II to do the motion control, and the camera was just a regular film camera?
JK: Yeah. What I was shooting was slit scan. It was a process I read about and was fascinated with and I wanted to try it. You really need a computer to control that stuff.

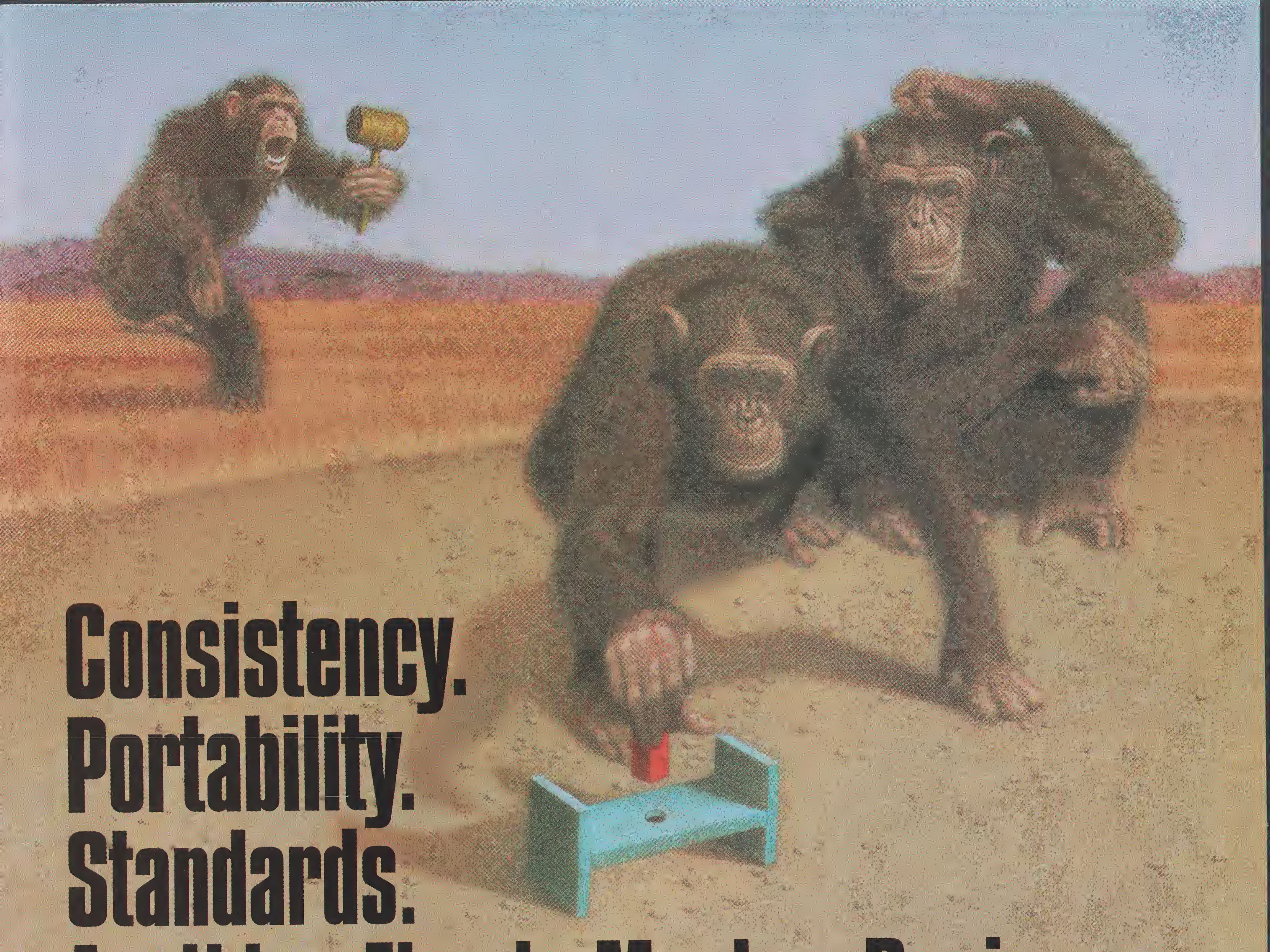
***DDJ*:** Did you write the software for the Apple II?
JK: Yes.

***DDJ*:** So you were familiar with programming at that time?
JK: A little. Actually, before I started at USC (in 1980), my dad got an Apple II as part of his university research work. After dinner, he'd go work on his research but he encouraged my brother Tom and I to play with it. This was in 1978.

The wonderful thing about the Apple II was it had this Basic interpreter built into ROM, so all you had to do was turn

---

Rick works for Sequoia Advanced Technologies. He can be contacted at thomas .tewell@seqadvtech.com.


Consistency.
Portability.
Standards.
Anything Else is Monkey Business.

(continued from page 34)
the computer on and start typing in lines. That was a lot of fun. I feel privileged that one of my first exposures to computers was when they were so simple. There was only so much that these really primitive computers could do, so it didn't take a lot to kind of learn everything there was to know about them. As the computers became more complicated, you could learn gradually. I can only imagine what it's like to dive into what programming is like now. I've had 20 years of exposure to it. Today, it's incredibly complicated for somebody just coming out of school.

***DDJ*:** At what point did it occur to you that the computer could actually be a tool for more than motion control or camera control—that the computer could actually be used to generate computer images suitable for film?

JK: A lot of people saw it coming. I read about computer graphics and had friends who were members of SIGGRAPH so I saw the tapes, and was fascinated by it. It wasn't really interesting enough to me at that point in the early '80s. I thought it was neat but not ready for feature films. But then as it started getting close to being ready, I became one of the first people pushing for it. I was computer graphics designer on *The Abyss* [circa 1989], which was one of the first realistic pieces of computer graphics in a feature film. At least that was our intent.

***DDJ*:** When did PhotoShop come into play?

JK: Actually, it was somewhat accidental. As I said, when I was a kid, one of my hobbies was model making. I got to be fairly good at that and it got me into the industry. But when model making turned into a profession, it sort of killed it as a hobby. It's not much fun to build models all day, then go home and build more models.

Since I was interested in motion control, I got a computer and started building motion-control systems for it. That became my new hobby. Because I knew people who were shooting motion-control elements with the models I was building, I began getting work as a camera assistant on motion-control stages. Then I got hired as a motion-control camera assistant at Industrial Light & Magic (ILM). Pretty soon I was doing motion control full time and its appeal as a hobby was greatly diminished.

I started at ILM in 1986 and had just gotten a Macintosh, my first sophisticated computer, and started writing little graphics programs as my new hobby. ILM was the first place I ever worked that had a computer-graphics department and, when I wasn't working in motion control, I'd go there to see what they were up to. They had this laser film scanner, where you could scan in a piece of negative and generate a digital image. They had the Pixar Image Computer, a nice high-quality frame buffer where you could do manipulations to a picture and film it back out. I had a demo of something so trivial now, you hardly even think of it. This guy brought up an image on the screen and simply sharpened it. That actually seemed miraculous at the time and made a huge impression on me.


Visual Effects Supervisor John Knoll (left) working with Senior Model Maker John Goodson (right) on a helicopter from Mission Impossible.

About that time, my brother Tom was at the University of Michigan working on his doctoral thesis. He had pursued computer programming much more seriously; that's what he had wanted to do for his career.

He was trying to get his doctorate in computer vision and the first part of any computer vision stuff is image processing. He was doing his thesis work on a Mac Plus and writing these image-processing algorithms as MPW shell tools. That was much like how Pixar Image Computers worked. You typed in command-line arguments from a UNIX command line to run C-shell scripts from the Sun to control the frame buffer on the Pixar. That was sort of the same thing Tom was doing on his Mac.

I saw a lot of the similarities. Then the Mac II came out. It had a math coprocessor. It had color. It was faster. It had more memory. I had to have it because I thought it was so neat. When that machine first came out, displaying a color image on it from a programming standpoint was a big deal. I wasn't terribly interested in the mechanics of the palette manager, window manager, and all the things that were required to display a color picture. What I was interested in was the code that figured out how bright a pixel should be. One of the hobby things I was doing was writing a little ray tracer. Tom told me to do the math, figure out how bright the pixel ought to be, and just write it to disk as a raw image. He said I could use his tools, which could read a raw block of bytes on the disk and display it as a picture and do various transforms to it.

I did this for a while, but it was cumbersome and I thought what would be neat was if we just built the display portion of this into an application so that I wouldn't have to fire up the whole MPW thing and run the shell tools to do this. One weekend, Tom spent a few hours bundling some of those functions in to this program called "Display." Once he had that working, I started bugging him for more stuff. It was like nothing was ever good enough. So we started adding more features until it struck me that we should sell this. We could get an ad in the back of *MacWorld* and sell it for 50 bucks. Tom was really skeptical.

***DDJ*:** Did you ever sell the product?

JK: No. Mostly what Display did was conversions. We had gotten it so that it could read several different image file formats. You could write several different image file formats and there were a couple of things you could do to them in the meantime. You could convert a color image to black and white.

I was completely full of naïve optimism. I showed it to a friend of mine at SuperMac, which was in alpha with a program called "PixelPaint." SuperMac was seriously considering making us an offer to bundle Display with PixelPaint as a file-format conversion utility. They had already run all their spreadsheets about how many units they thought they were going to sell of PixelPaint and what kind of deal would they want to make with us on bundling this. That added up to a number that seemed like this was worth doing.

I called Tom and said SuperMac was interested, so he scheduled two days a week to work on it full time. After two or three months, it really did a lot of things. It didn't really fit in my mind as utility any more. It was a program in its own right that wanted to be sold as its own product. One day I called Tom up and told him that I didn't think there would ever be an opportunity like this like thrown at our feet again. We just had to drop everything to make this happen.

Tom estimated he was six months from finishing his doctoral thesis. In a supreme act of faith, he stopped working on his thesis and started programming full time. We greatly underestimated how much work this was going to be. When Tom

(continued from page 36)
stopped school, he figured he had about six months of programming and we'd wrap up Version 1.0 of this program, and he could start next semester and finish his thesis. Meanwhile, we'd be making some money.

From the time he decided to stop school until Version 1.0 shipped was almost two years. It became much bigger than we thought it would, but it kept getting better and better. Tom is really a superb programmer. He's one of the best engineers I know. He just wrote this terrific, great code.

At the time, I moved from motion control over to computer graphics, so I was doing a lot of work on the Pixar Image Computers—running composites and doing image-processing scripts. That drove a lot of my input as to what kind of features ought to be in PhotoShop. I would try to do more and more of my work in PhotoShop and try stuff. That's sort of how "feathering" got born. It was actually me using it for little projects that helped define the feature set.

Version 1.0 was a usable tool largely because I was trying to use it to solve real-world problems. I would run into something that would just stymie me. There's got to be a way of doing this, and then Tom would scratch his head and go, "That would be hard." He would think about it for a while. I would talk to him a few days later and he would say, "I was thinking about that and I had this great idea."

I was goading him a little bit, too. I would say, "You know what I really want to do? I want to make one of these selections so that I can like select some area and then the paint only affects just the area selected." Tom would say, "Oh, that's going to be impossible to make that go real time. It's going to be really slow." I'd say "Oh, come on, Tom. I'll bet you can do that." About a week later he would say, "I was thinking about it, and I think I've got a way." It was often a whole lot of exchanges like that where at first Tom thought it would be really hard, but he would keep thinking about it. He's brilliant that way, and he would come up with a clever solution to the problem.

***DDJ*:** When PhotoShop was born, the industry was in some interesting transitions in computer graphics.

JK: Yes. We started on PhotoShop in September of 1987. I think 1.0 shipped in January of 1990. There was some time between when we started and when it shipped. A lot of things happened in that time. I started working in computer graphics…it wasn't until late 1988, I think. The first thing I did in computer graphics was a Pacific Bell Smart Yellow Pages commercial.

***DDJ*:** With a Pixar?

JK: Yeah. A Pixar Image Computer is basically a frame buffer. Lucas Film Computer Division was working on what became the Pixar computer. "Pixar" adopted that name as the name of the company after George [Lucas] sold it to Steve Jobs.

***DDJ*:** So that was something that was invented and not available anywhere else except for here?

JK: Right. We had two of them here that we used for composite work and various image-processing things. On all the old Pixar films like *Andre and Wally B* [circa 1984], they would render different parts of the shot as separate passes so the character in the foreground would be rendered separate from the background. Then they would composite them together, and the tool they used to do it would be the Pixar Image Computer.

> *We try to use off-the-shelf software wherever we can*

***DDJ*:** When *The Abyss* was created, what was the state of computer graphics?

JK: In general, no one thought of computer graphics as something you could use for real on a feature film to do something that looked realistic. The one exception was the stained-glass man [from *Young Sherlock Holmes*, circa 1985], which was a pretty remarkable achievement, and it's the only thing that had ever been quite like that to that point. Stuff like *Last Star Fighter* [circa 1984], nobody really considered realistic. But I was impressed with stained-glass man because it had things like depth of field.

Right after I started, our computer-graphics department had done this *Star Trek IV* [circa 1986] dream sequence with the floating heads. It didn't look very realistic. It was intended to be a stylized thing. I don't know if anybody thought that our tools in house were ready to do something super realistic.

I remember we got the storyboards on *The Abyss*, they were these beautiful shaded drawings. They are really fascinating. The imagery was really neat. "Wow, these are going to be really cool shots—whoever does this and however it gets done." A lot of different approaches were being bandied about with things even as weird as stop-motion animation with clay with images of water projected onto it. Things that almost certainly never would have worked.

We had just gotten an SGI with Alias, and Jay Riddle in the computer-graphics department did a little test making some sort of a water tentacle thing. It was not a sophisticated test, but he did it really quick. He did it, I think, overnight and showed it to Jim Cameron [*Titanic* writer/director] the next day. Jim was really surprised how quickly that had been done because the reputation was that computer graphics was really, really slow and very expensive and the complete antithesis of interactivity. You'd talk to these guys and they'd disappear for months, and then they would come back with something you didn't want. "I want it to be more like…" "Well that will be another six months."

***DDJ*:** But they felt this was an intricate part of the film?

JK: Jim's position was that if the water tentacle sequence—while it was a bold thing to attempt—didn't work or ended up looking terrible, he could cut it out of the movie and he could still make the movie. He wasn't hinging the success of this picture on this effect working. It was only like 25 shots. This seemed like a huge number to us at the time, but it is hardly anything now. So we started this R&D project to do this thing, and we wrote a bunch of new software to do it. We switched over from Rays to RenderMan, which Pixar had just gotten going.

***DDJ*:** There was nothing on the street that could do this at the time?

JK: No. We used the RenderMan renderer but we wrote custom shaders to do the fake refraction and get the right amount of reflection for fog and that sort of thing. We had to write the software for it to do the rippling of the surface and to "skin" it. The way it was actually done was, we animated a spline in space—a 3D path—and we had a bunch of cross sections. They were animated separately, so it was just a bunch of circles, and we scaled them. And then, there was a piece of software called "Skin" that would take all of the circular cross sections and place them perpendicular to the spine at particular points and skin the surface.

Then there was another program that would let you place a bunch of 3D noise generators in the world, and it would take the patches and subdivide them into smaller patches and perturb all the control

(continued from page 38)
vertices according to the sums of all the sine waves from 3D noise generators. So the model was created new per frame based on this program, so some work was involved. How do you do motion blurs when you're actually just changing the model from frame to frame without taking one model and moving it? Some hacks were made. Actually it's the same model, but what we're doing is we're moving these vertices from here to here. You would write two-root files. They contain all the same CVs [control vertices] and then there was a script called JR2R that would take the two-root files and make them look like it was one model just moving from this frame to that frame.

***DDJ*:** Then comes *Terminator 2* [circa 1991], which has something (not quite like the water tentacle) but it has the Mercury guy and that was from James Cameron.
JK: Yeah. Jim said it was a big gamble. If it didn't work, he could always cut it out of the picture, but based on his experience on *The Abyss,* he went much bolder on *Terminator 2* with making a character that had to be done with computer graphics. And it had to work because if you cut that out of the movie, you've got nothing left. All the things of being able to change shape from this to that and to melt and then reform itself. Well, the effect has to work or you don't have a movie. Yeah, it was a sign of his faith in the technology.

***DDJ*:** In *Jumanji* [circa 1995] we have the first computer graphics hair that actually flows and moves, and the depth is there, and it is so stunningly realistic that it was actually an amazing achievement for computer graphics. Did that require custom tools or was there a point where you could actually use off-the-shelf components to actually do this?
JK: We try to use off-the-shelf software wherever we can, but a lot of things we're called upon to do just can't be done with off-the-shelf software. So we have a pretty good size software-development staff just to develop these tools; otherwise, we would just have to say, "No, we can't do that."

***DDJ*:** Do you still do that today?
JK: Yeah.

***DDJ*:** Do producers come in and say, "We want to produce a film and here are the special effects that we want" and you just go, "I don't think so."
JK: Well, no. We gulp and say, "Okay, we can do that. Here's the budget." Then they gulp.

You can usually spend your way out of just about any hole there is. If you put enough time and man hours into something, there's usually a way to do it and I can think of very few exceptions where we just have to give up and say, "No, that just can't be done." There are some things that would be extremely difficult and we could never do realistically, at least not yet. But most of the things we're asked to do are at least within some amount of R&D of what we're capable of. George [Lucas], on this new *Star Wars* picture, wrote a lot of things into the script without worrying about how the hell are we going to do this. He just writes things he thinks are neat.

***DDJ*:** Martin Hash has created a product called Animation Master and is trying to make a film, *Telepresence,* for $2 million which positively could not be made for $2 million if a studio did it based on the effects he wants to put in there.

Do you see a trend coming where independent filmmakers can use off-the-shelf components to actually have "big budget" special effects in films? Up to now, independent films have been pretty much lacking special effects that are just sort of character driven.
JK: It's already happening. A bunch of friends of mine are starting up these garage operations—little one-man digital facilities—and they do things for TV shows or low-budget features. They're able to do the kind of work now just at home with PCs. It used to be that you had to have the whole full-blown production mechanism here for it, and now you can do some pretty good looking stuff.

***DDJ*:** Like Electric Image?
JK: Yeah. With Electric Image, After Effects, and PhotoShop, you've got a little production facility there.

***DDJ*:** Speaking to a peer programming audience, what do you see as the next generation of products for computer graphics?
JK: Well, I don't think there are any real specifics that are easy to predict. But I think the general trend is to try and eliminate as much machinery between the artist and the art as possible.

One of the things that has been really liberating about moving to digital-production techniques is that it used to be that huge amounts of effort went into just the mechanics of not getting the matte line or not getting the wrong color in a shot, for instance. That's where a lot of your energy went—just trying to get rid of the really

(continued from page 40)

obvious things. Now, you can take that stuff more for granted. Today, an artist spends more time working on the aspects of the work that make the shot look good or not look good, and not so much on the mechanical. I see that trend continuing.

Right now, my biggest complaint about the way that a lot of these digital tools work is that they're still kind of awkward, and the artist spends too much time working on things that have nothing to do with the shot looking good or not. It's editing exclusion lists and making sure your aliases are pointing to the right directories. There's a lot of machinery that the artist still has to deal with, that, as software gets better, they're going to spend less of their time of doing and more of their time focused on the real art of it.

***DDJ*:** What about these new digital interfaces like FireWire? Do you see that again liberating artists so that digital images can go straight into the machine?

JK: I think that all these technologies like this are wonderful. I spend a lot of my time living on the bleeding edge, where we're just trying to get something done almost no matter how painful it is. We work with these kind of kludgy custom-written things that just barely work well enough to get through the shot or you really wouldn't want to do that a whole lot more. And what happens is that like five years down the line, the commercial applications end up with a lot of functionality that we have very painstakingly hand crafted—like morphing, for example.

Back on *Willow* [circa 1988], Doug Smythe spent time writing the first morphing program that worked well for what we did and let us do these shots that were sort of impossible otherwise. We made good use of it. I used it on *The Abyss* to actually do the face animation with morphing. We used it on *Terminator 2*. Then Elastic Reality hit the market and once that capability was present in the commercial program, it was at least as good as our morph program. In some ways, it was better, and there was no reason to keep working on our program.

A commercial application now had the same functionality, and you could buy it for nothing. That's a good example of something that we sort of suffer through getting the first version, and then people see the results of that, and they go, "Oh, man, I want this." So a bunch of commercial developers can jump in and say, "Well we can provide that." They write a good interface on it, on something that's actually debugged with appropriate error messages and all those kinds of things that commercial software brings to the equation. And then it's available to everybody.

DDJ

Build on a

Rock Solid Test Foundation

LabWindows™/CVI

Virtual Instrumentation Tools for C/C++

When you need to build a BIG test system...

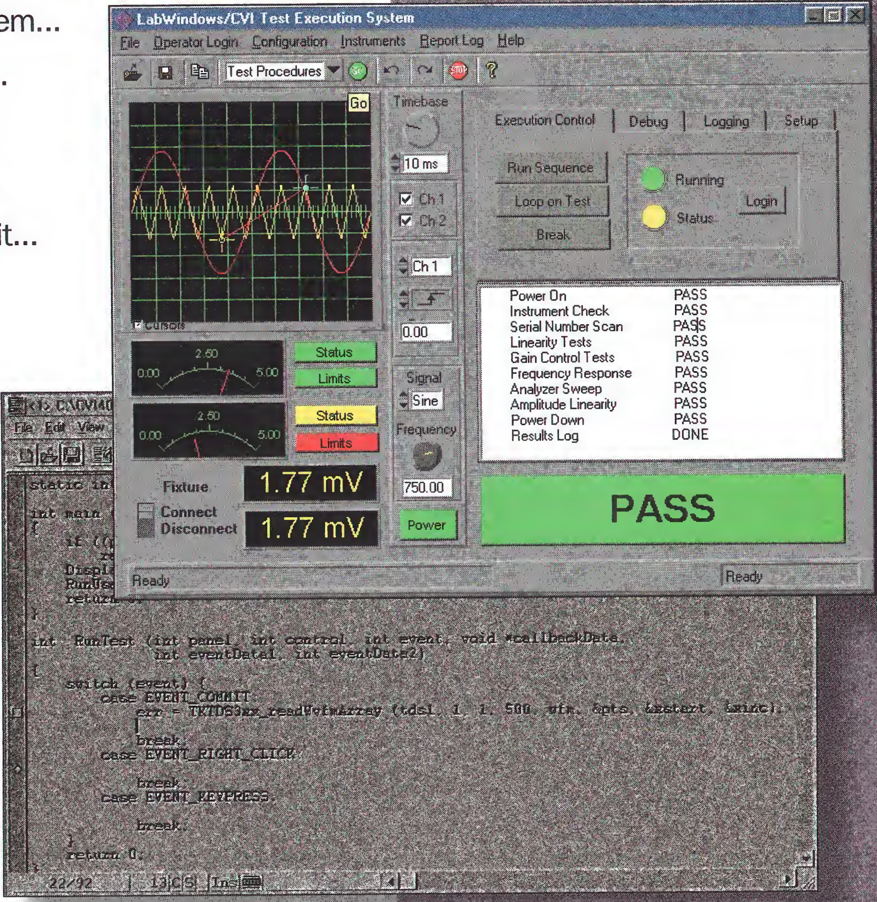
When you can't afford to get locked in...

When it has to be fast...

When you need a solution that works...

When your production line depends on it...

When your job is on the line...

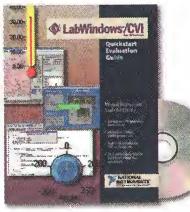


Why look anywhere else?

- ✔ Standard ANSI C Programming
- ✔ Lightning-fast code generation
- ✔ Open development environment
- ✔ VXI*plug&play* driver standard

For data acquisition, GPIB, and VXI instrument control, analysis, and user interface tools, LabWindows/CVI has everything you need under one roof.

Put LabWindows/CVI to Your Test!



Call today for your FREE LabWindows/CVI Evaluation Version

(800) 661-6063

(U.S. and Canada)

NATIONAL INSTRUMENTS™

The Software is the Instrument™

U.S. Corporate Headquarters

Tel: (512) 794-0100 • Fax: (512) 794-8411

info@natinst.com • www.natinst.com

Worldwide network of direct offices and distributors.

A Windows 3D Model Viewer for OpenGL

Combining Win32 with OpenGL

Jawed Karim

OpenGL is known in the UNIX world as the 3D API behind high-powered scientific applications. It has recently gained attention in the PC sector, thanks to the computer-game industry, which has embraced OpenGL as an API standard for 3D game programming. Furthermore, 3D hardware acceleration for PCs has extended the range of applications for OpenGL even further.

The OpenGL API is intuitive, easier to use, in my opinion, than Microsoft's Direct3D API, and is portable among platforms. In this article, I'll present a model viewer for use with OpenGL on Windows 95/NT. First, however, I'll describe the important parts of a Quake2 model viewer—an OpenGL-based system written in C/C++—that displays wire-frame and texture-mapped models (see Figure 1) from Quake2 and provides a basic interface to modify their appearance. In the process, I'll focus on file formats (MD2 files for models, and PCX files for textures), passing the data contained in the files to OpenGL for rendering, and interfacing Win32 with OpenGL using an API called "WGL." The archive Q2M-SRC.ZIP contains the Quake2 Model Viewer source code, while Q2M-BIN.ZIP is the Quake2 Model Viewer EXE file. Both are available electronically; see "Resource Center," page 3.

Jawed studies computer science at the University of Illinois at Urbana-Champaign. He works part-time at the National Center for Supercomputing Applications, and can be contacted at jkarim@students.uiuc.edu.

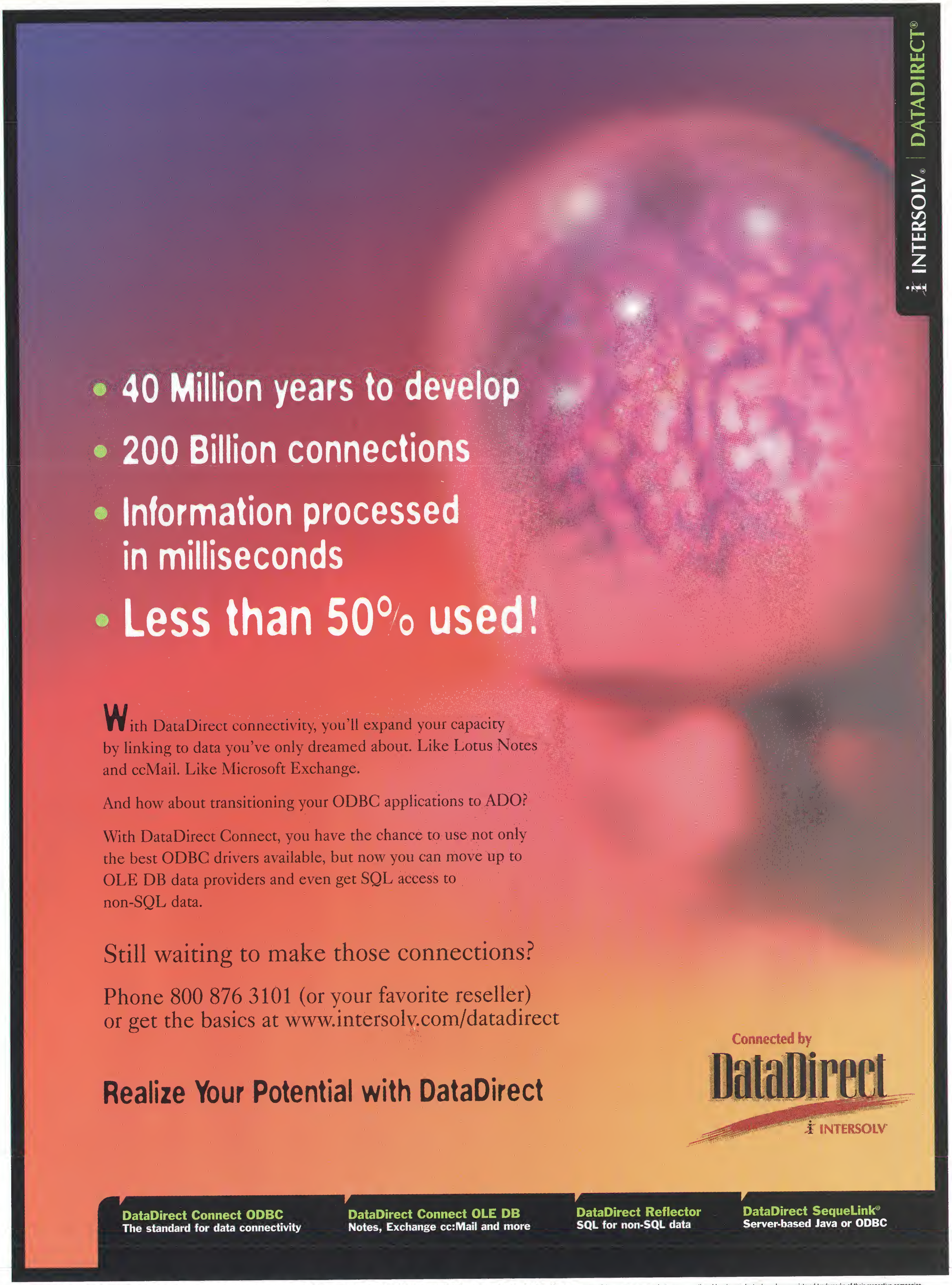
Reading the MD2 File Format

The only official source of information about Quake2's MD2 format is code by John Carmack of id Software; this code writes 3D polygon mesh data to an MD2 file (available at ftp://ftp.idsoftware.com/). Anyone who has looked at this source code will notice that some of the *structs* in Quake2 Model Viewer's md2.h (available electronically) are derived from it. Writing the MD2 reader basically involves converting John's code from reading MD2 files to writing them. Figure 2 illustrates the binary structure of an MD2 file.

To display the textured Quake2 models, four specific types of information are needed (see Figure 3):

- 3D vertex coordinates.
- A list of triangles consisting of those vertices.
- 2D texture vertex coordinates (one for each 3D vertex).
- The texture image.

All of the 3D vertices in the model are stored in one array. When the triangles (which are made up of those vertices) are defined, all that has to be stored for each vertex of a triangle is an index number to the big vertex array. The reason for this is simple: Since many of the vertices are shared between triangles, storing each vertex once saves memory. In addition, linear transformations can be



INTERSOLV® | DATADIRECT®
• 40 Million years to develop
• 200 Billion connections
• Information processed in milliseconds
• Less than 50% used!
With DataDirect connectivity, you'll expand your capacity by linking to data you've only dreamed about. Like Lotus Notes and ccMail. Like Microsoft Exchange.
And how about transitioning your ODBC applications to ADO?
With DataDirect Connect, you have the chance to use not only the best ODBC drivers available, but now you can move up to OLE DB data providers and even get SQL access to non-SQL data.
Still waiting to make those connections?
Phone 800 876 3101 (or your favorite reseller) or get the basics at www.intersolv.com/datadirect
Realize Your Potential with DataDirect
Connected by
DataDirect
INTERSOLV
DataDirect Connect ODBC
The standard for data connectivity
DataDirect Connect OLE DB
Notes, Exchange cc:Mail and more
DataDirect Reflector
SQL for non-SQL data
DataDirect SequeLink®
Server-based Java or ODBC

(continued from page 44)
performed on the entire array at once, thereby speeding rendering time. Since the texture image itself is not a part of the MD2 file, it can be read in from a conventional PCX file.

Before starting, you must know how much data to expect. The file's header section tells you the number of vertices, triangles, and texture coordinates contained in the file. Knowing when to stop, you can go into a loop and read the information in chunks. To store all the data, use the vertex structure in Listing One (listings begin on page 96).

Each triangle is defined by its corners, *a*, *b*, and *c*. These values are indices to an array of type *make_vertex_list*, which is a list of all 3D vertices in the entire model. The remaining six integers represent the 2D texture coordinates for every vertex. Listing Two is an example of a structure for holding this data. Using such a structure, the coordinates of the three vertices of the first triangle in the model can be referenced (see Listing Three).

In a Quake2 model, the only things that differ from one frame to the next are the 3D coordinates of the triangle vertices; the vertex indices and texture coordinates remain the same. From frame to frame, each triangle still consists of the same three vertices — only the vertices undergo linear transformations. To hold each frame in an array, you create another array of type *make_frame_list* (Listing Four), each of

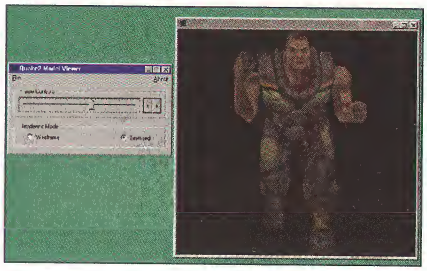


Figure 1: *The 3D model viewer in action.*

which contains an array of vertex coordinates (Vertex 1, 2, and 3, respectively). There exists one copy of this array for each frame. Having filled all of the data structures, you can look up the coordinates of any polygon in any frame; see Listing Five (the coordinates of polygon P in frame F).

Texturing the Object

Quake2's model textures reside as separate PCX files, either in the pak0.pak file or quake2/baseq2 directory. Since OpenGL itself does not provide a way to read the binary PCX graphics file format, you can read the PCX file and pass its data to OpenGL.

Figure 4 describes the PCX format. The three basic sections in the file are the header, pixel data, and palette data. You can use two arrays of type *unsigned char* to store the last two sections. The header contains some basic information about the particular file, such as the PCX version, and the file dimensions. If the file is actually a PCX Version 5 file, the first two bytes in the file must be equal to 10 and 5, respectively. Having determined the image dimensions from the header section, you dynamically allocate an array of type *unsigned char* of *size(width*height)* for the pixel data and read it into the buffer byte-by-byte. Because a Version 5 PCX file can support exactly 256 colors, the size of the palette section is always 768 bytes (3*256, or RBG*256).

When the *CImage::Read (char filename[])* function is finished, the *m_pixel_buffer* array is filled with all the pixels in the image, and *m_palette_buffer* contains consecutive RGB values for each of the colors.

How do you get the color of a specific pixel in the image? The pixel buffer simply contains index values of the palette buffer. Listing Six shows two methods. The R, G, and B components of the first pixel (pixel zero) in the image are Listing Six(a). However, because the palette array contains consecutive RBG values (RGBRGBRGBRGB...) for all the colors, the individual R, G, and B values at pixel position P are obtained by properly offsetting the array index; see Listing Six(b).

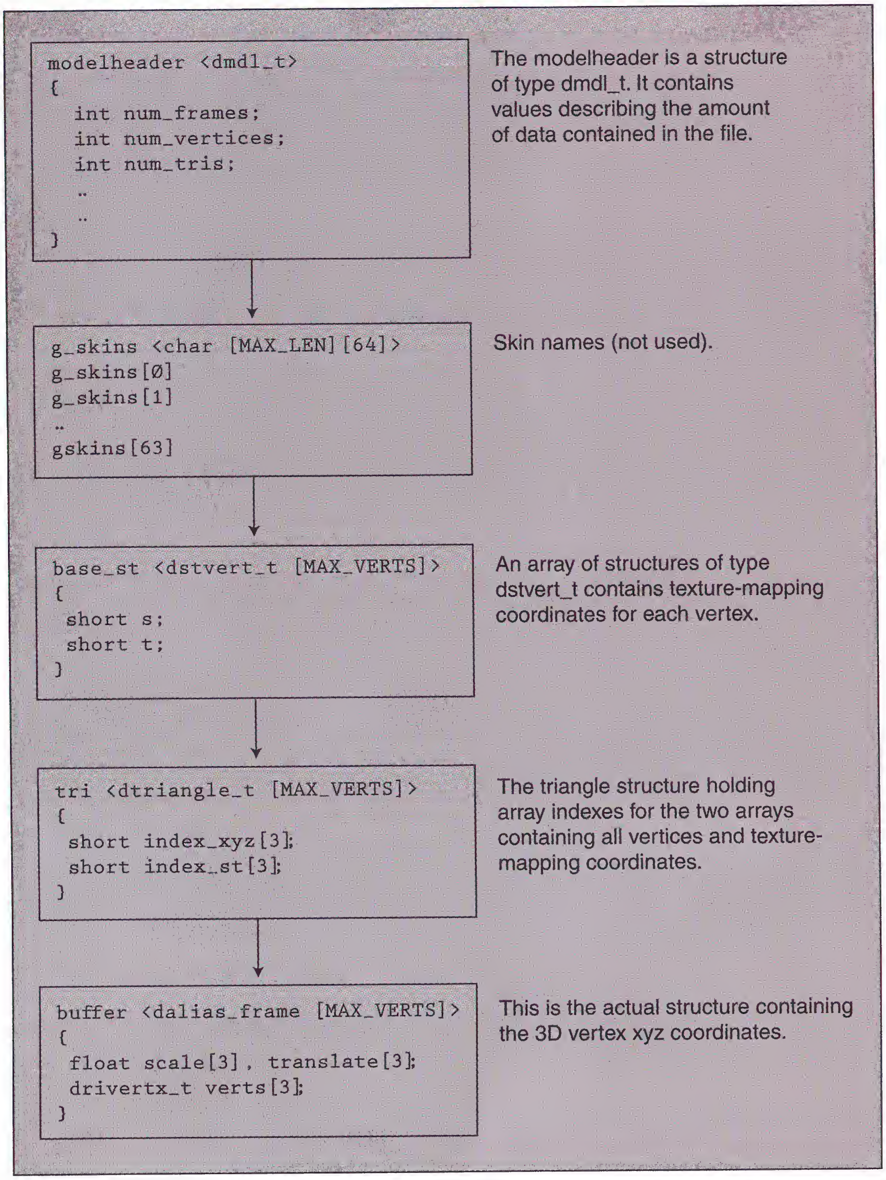


***Figure 2:** The binary structure of an MD2 file.*

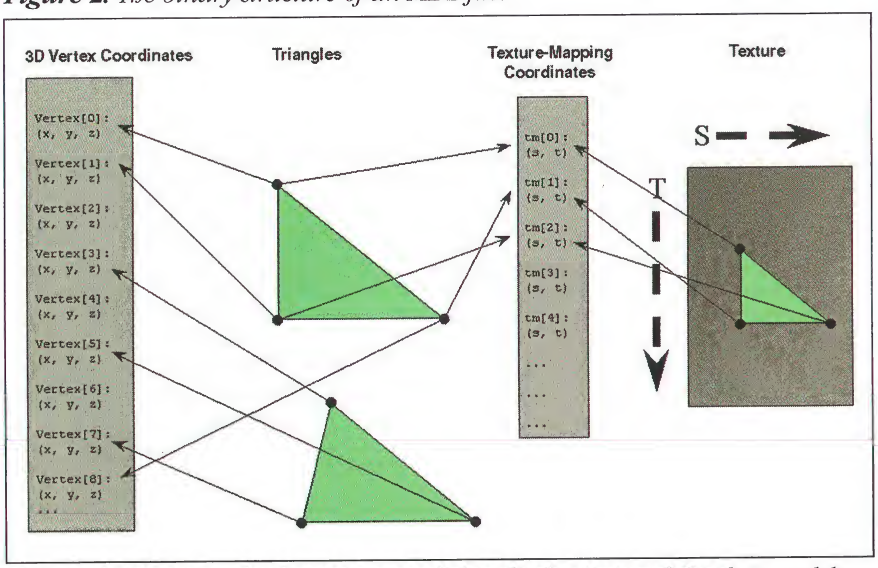


***Figure 3:** The types of information needed to display textured Quake2 models.*

Finally, to be able to reference color values at specific (X,Y) coordinates in the texture, *P* is substituted by *X+Y*Width*, where *Width* is the width of the texture; see Listing Six(c).

OpenGL

Once the necessary data is organized and stored in memory, you can start rendering using OpenGL. But first, some of OpenGL's texturing options must be set. In particular, you must specify how to treat textures when wrapped and indicate the "minification" and magnification filters (Listing Seven).

In addition, back-face culling and texturing have to be explicitly enabled. Since you won't be looking at the backsides of polygons, you only have to enable front-side filling of polygons. Lastly, you specify the texture function (Listing Eight).

OpenGL's *glTexImage2D()* is the function that actually textures the object. It expects to be passed, among other parameters, a pointer to an array containing successive RGBA values for each pixel in the texture (for example, RGBARGBA-RGBA...).

Thus, before calling *glTexImage2D()*, two changes must be made:

1. The pixel and palette data read from the PCX file must be copied into another array, of a format that *glTexImage2D()* can accept as a parameter.
2. Because OpenGL requires the dimensions of a texture to be powers of two, the texture has to be rescaled first using *gluScaleImage()*.

Both of these steps are accomplished in *CImage::Image2GLTexture()*, which first creates a new array called *unScaled*, fills it with RGBA components, and rescales it to an appropriate size. The loop in Listing Nine fills a new array with RGBA components of each pixel in the texture, again offsetting the array indices as in the PCX code.

WGL provides an interface between the Win32 API and OpenGL

Now the texture contained within *unScaled* can be rescaled to have dimensions that are powers of two. To prevent the texture from losing much quality while keeping the performance at a reasonable level, a power of two that is closest to the original dimension will be used. For example, if the original width is greater than 256 pixels, the new dimension should be 512 pixels. If the original width is 128 or greater (but less than 256), the rescaled dimension should be 256. After a series of *if* statements have determined a good fit for the new dimensions, a call to *gluScaleImage()* rescales the texture (Listing Ten).

Finally, the *glTexture* array can be passed to OpenGL as follows: *glTexImage2D(GL_TEXTURE_2D,0,4,scaledWidth, scaledHeight,0,GL_RGBA,GL_UNSIGNED_BYTE, glTexture);*. Table 1 provides a quick explanation of the parameters.

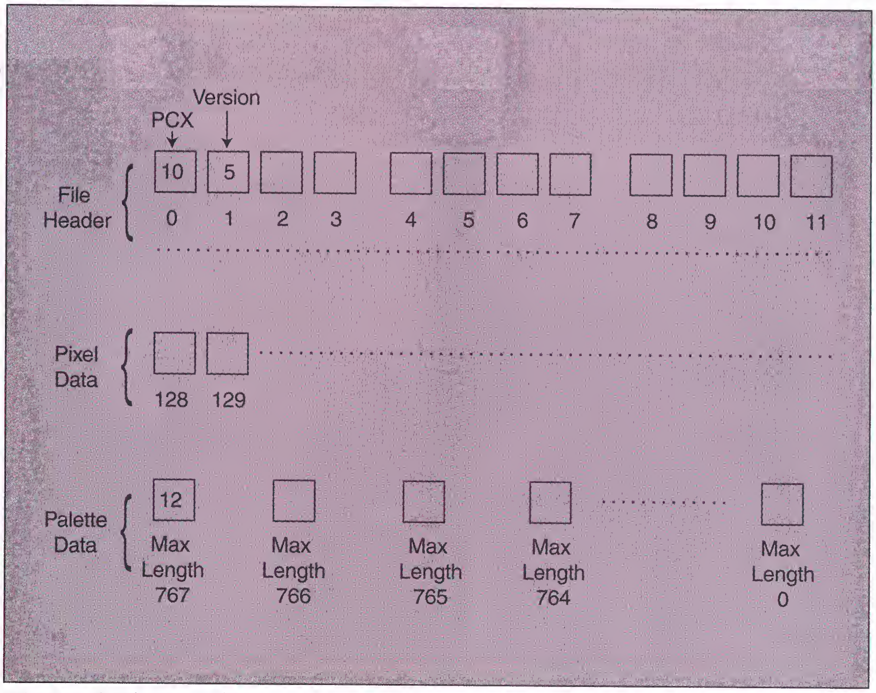


Figure 4: *The PCX format.*

Creating an OpenGL Rendering Context

WGL provides an interface between the Win32 API and OpenGL. It sets up a palette for your rendering window and handles such things as double buffering. To do this, you usually need to use four or five of the fewer than 20 WGL functions. I have written a basic C++ wrapper class for the functions that is easy to use. Most of the code in the *COpenGLWindow* class is taken from Silicon Graphics' OpenGL Developer Tools CD-ROM for Windows 95/NT, which interestingly has become a collector's item since SGI's "Fahrenheit" deal with Microsoft. (SGI is cooperating with Microsoft on the next generation of OpenGL. Since the agreement, SGI's, OpenGL drivers for Windows 95/NT have disappeared from the SGI web site, and the SGI OpenGL Developer CD-ROM for Windows 95/NT is hard to come by. However, there are several web sites mirroring its contents, including http://jawed .ncsa.uiuc.edu/.)

The dimensions of the rendering window are passed to the constructor, but its window handle must be passed to the *OpenGLWindow::Create()* class member function to actually create the rendering context.

WGL does not physically create a window for you; that is Win32's responsibility. WGL creates an OpenGL rendering context for a window that has already been created. If you want a window to create and destroy its OpenGL rendering context as the window is created and destroyed, simply catch the WM_CREATE and WM_DESTROY messages in the window's window procedure. Then call *OpenGLWindow::Create()* and *OpenGLWindow::Destroy()*, respectively, as has been done in inter.c's *GraphicsProc* function (available electronically). The only other time you really need to use WGL is for a system palette change. Windows will indicate that such a change has been made by sending a WM_PALETTECHANGED message to every window, and then *OpenGLWindow::RedoPalette()* will take care of the change.

Drawing the Entire Model

Inter.cpp's *redraw()* function (available electronically) redraws the entire model in its current state by specifying all of the triangle vertex coordinates and texture mapping coordinates between *glBegin(GL_TRIANGLES)* and *glEnd()*. This requires three calls to *glTexCoord2f()* (two parameters) and *glVertex3f()* (three parameters) for every triangle. One thing to note about the *glTextCoord2f()* function is that OpenGL expects texture-mapping coordinates to be relative, not absolute. To obtain these coordinate values, divide the original texture mapping coordinates from the model by their maximum range in the texture. In other words, divide the *S* component by the texture map's width and divide *T* by the texture map's height. These values will fall between 0 and 1 and remain unchanged when the texture is resized. For instance, (0.5, 0.5) will always point to the center pixel of the texture, no matter whether the texture dimensions are 173×233 or 256×256. Of course, doing a floating-point divide three times per loop is inefficient. By storing these values ahead of time the loop's efficiency could be improved greatly.

Between frame redraws the rendering window's window procedure keeps track of mouse movements and mouse button activity by listening to WM_MOUSEMOVE, and WM_*BUTTON(UP/DOWN) messages. The movement increments are then temporarily stored in two arrays— one for translational movements, and another one for rotations. At the beginning of each frame redraw the linear transformations are carried out using *glTranslate()* and *glRotate()*.

Code	Definition
GL_TEXTURE_2D	Defines a two-dimensional texture.
0	Supplies one texture as multiple resolutions.
4	Indicates which of the R, G, B, and A values are used.
scaledWidth	New width.
scaledHeight	New height.
0	Width of the border (no border).
GL_RGBA	Format of the texture data.
GL_UNSIGNED_BYTE	Data type of the texture data.
glTexture	Pointer to array containing texture to be rescaled.

Table 1: *Explanation of the parameters in* glTexImage2D(GL_TEXTURE_2D,0, 4,scaledWidth, scaledHeight,0,GL_RGBA,GL_UNSIGNED_BYTE, glTexture);.

Conclusion

Although OpenGL is straightforward to use, simply knowing the API is not sufficient. Since OpenGL does not provide functions to read 3D model and texture files of your preferred format, a basic understanding of 3D concepts and some amount of manual data manipulation is also required. Combining Win32 with OpenGL makes it possible to develop applications with user-friendly interfaces and impressive 3D graphics.

Keep in mind that one of OpenGL's bonuses is portability. Porting your Win32 OpenGL applications to X under UNIX should not be much more difficult than cutting and pasting some of the graphics code. Of course, creating another interface from scratch will be necessary.

DDJ

(Listings begin on page 96.)

The Kernel Graphics Interface

A portable high-performance graphics subsystem

Andreas Beck

Determining how an operating system should handle graphics is an exercise in tradeoffs. If you are interested in the fastest possible graphics performance, the only solution is for your application to work directly with the graphics hardware without regard to security. However, if you are willing to sacrifice a little bit of speed to gain portability and a degree of safety, GGI could help you a lot.

The GGI (General Graphics Interface) project (http://www.ggi-project.org/) is intended to bring safe, fast, and portable graphics to a variety of platforms and operating systems. GGI consists of user-level libraries of basic graphics functions and kernel-level drivers that handle the low-level graphics routines. The Kernel Graphics Interface (KGI) is the kernel console interface upon which the Linux implementation of GGI is based. Figure 1 shows how GGI and KGI are related. In this article, I describe the motivation, architecture, and implementation of KGI.

GGI is not confined to Linux, nor to KGI as the display subsystem. LibGGI is a lightweight graphics library that runs on a variety of platforms and graphics subsystems like X-Windows (tested on Solaris, AIX, IRIX, Linux, and others), SVGAlib (Linux), or other native graphics interfaces like the Sun framebuffer device. Ports for more targets (such as Microsoft Windows) are in the works.

Andreas studies physics at the University of Düsseldorf, Germany. He can be reached at andreas.beck@ggi-project.org.

The Problem

The job of an operating system is to arbitrate access to hardware to preserve the stability of the system, prevent software from damaging the hardware, and provide the software with an abstracted view of the hardware.

Few operating systems do this properly for graphics cards. Graphics support is either placed entirely in the kernel (like NT) or is left to user-mode applications with special permissions (like traditional Linux SVGAlib or X applications).

From a security point of view, there is nothing wrong with placing all graphics functionality in the kernel. The problem is that it vastly increases the kernel size at the expense of stability. Video drivers become more difficult to write and especially to debug—and errors in the drivers impact system stability.

On the other hand, the SUID root approach used by X and SVGAlib presents some security hazards. In general, you want to avoid running any applications as SUID root, since buggy or malicious code can easily be manipulated to break into, or simply break, a system.

A malicious, or merely carelessly programmed, graphics application can easily hang the system by causing a bus lockup (possible with many graphics cards due to bad programming), leaving the console in graphics mode (making it hard to use the system), or locking out virtual console switching. Worst of all, a malicious application might even be capable of damaging hardware by programming unsuitable clocks, thus overloading the RAMDAC and/or monitor. While most modern monitors have protection circuitry for this, RAMDACs are usually without defense.

X circumvents this problem somewhat by being a client-server system, which protects the privileged server from malicious or buggy user code. Yet even then, it is still possible to abuse the X server, for instance, to read any file on your system (see http://www.rootshell.com/).

SVGAlib is a bigger problem, because its applications must be SUID root. Consider the binary-only releases that are necessary for commercial games but must run SUID root. Would you trust all vendors not to spy on your system? Would you always check PGP signatures to make sure you don't have a hacked copy with some Trojan Horse? Even worse, normal users can't develop SVGAlib applications since root access is necessary to give appropriate permissions to the executable so it can be tested.

The Solution

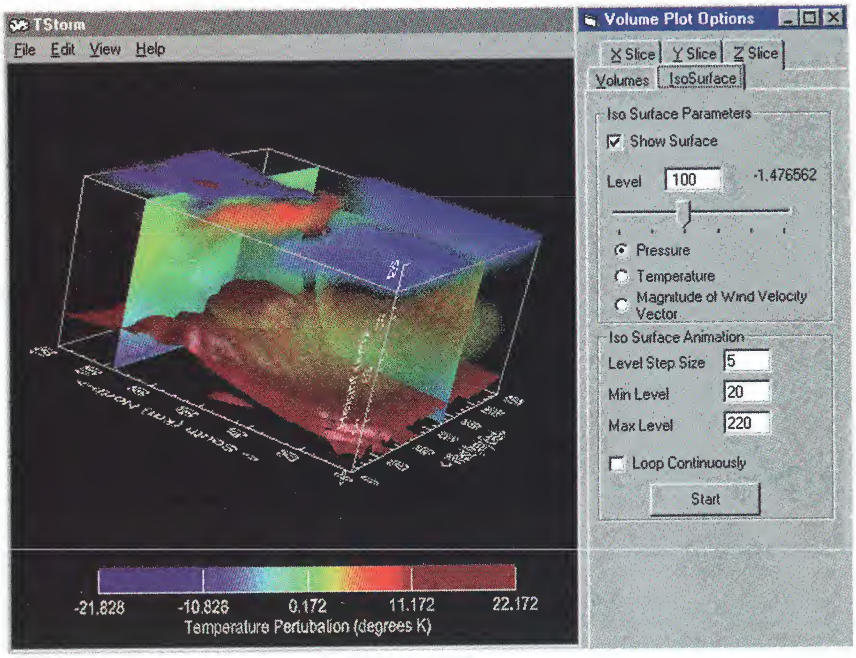
KGI tries to address these problems by moving only the critical part—the actual programming of the graphics hardware—to the kernel. This reduces the security problems to those that any UNIX device

NOW INCLUDES
ACTIVEX CONTROL

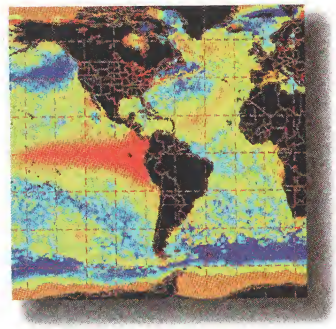
ADVANCED VISUALIZATION, RIGHT IN MY NATIVE PROGRAMMING ENVIRONMENT?

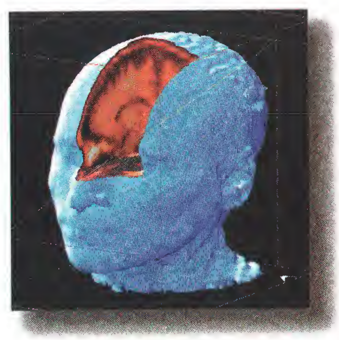
You bet. With IDL' s ActiveX Control you'll visualize your data in ways you've never imagined. You'll get state-of-the-art rendering. High performance data analysis. Advanced algorithms. All within your native programming environment.

We've been developing IDL, the *Interactive Data Language*, for over 20 years and have customers in the world's largest medical, engineering, science and military organizations. Once you see how easy IDL is for solving big visualization problems, you'll wonder why you waited so long.



Tornadic thunderstorm simulation using IDL's ActiveX control and OpenGL accelerated graphics. Image courtesy of Joe Klemp, Ph.D., NCAR.





Multiple images of the Sun recorded via a CCD camera/Soft X-ray Telescope. Image courtesy of Lockheed's Solar and Astrophysics Laboratory.

El Nino's effects are studied by analyzing and visualizing sea surface temperature data in IDL. Data and image, from Oct. 14, 1997, are courtesy of NOAA/NESDIS.

Medical researchers can investigate and display 3D MRI data with IDL's slicer tool.

IDL®

The Interactive Data Language

Download IDL at www.rsinc.com today or send in for a Free demo CD.

Research Systems, Inc. tel: 303.786.9900 email: info@rsinc.com **Software ≡ Vision**

International Offices: Austria, Germany, Liechtenstein, Luxembourg, Switzerland, The Netherlands CREASO, GmbH tel: 49 8105 25055 • **Brazil** SulSoft tel: 55 51 337 38 91 • **China** 3 - Link Systems Pte Ltd. tel: 86 10 6217 9910 • **France, Belgium** Research Systems International, France SARL tel: 33 1 34 58 72 40 • **India** Sierra Optima Limited tel: 91 40 3740368 **Italy** Research Systems Italia S.R.L. tel: 39 39 605 8605 • **Japan** Adam Net Ltd. tel: 81 35802 2251 • **Korea** InterSys tel: 82 42 862 8100 • **Spain, Portugal** Estudio Atlas tel: 34 945 298 080 **Taiwan** Concentrate Corporation tel: 886 2 883 7752 • **United Kingdom** Floating Point Systems UK Ltd. tel: 44 118 977 6333

(continued from page 50)
exposes: inappropriate file system permissions and bugs in the driver.

KGI does not do the actual drawing in the kernel. It's not necessary, and doing so would increase the possibility of errors that are even more serious when they happen in a kernel context. The KGI driver is designed to be a thin layer around the hardware functionality. It only abstracts functions that are fairly standard between different cards.

Functions for setting up modes and some common accelerated drawing functions are available via a standard command API, while card-specific quirks are exported in a private command area that is called by a card-specific user-mode counterpart.

Implementation Considerations

Speed is the main problem with a graphics interface that is at least partially running in kernel mode. If you needed to make a kernel mode call every time you called a basic function like drawing a pixel, the system would crawl.

Fortunately, almost all available cards have some notion of a framebuffer, a portion of the onboard Video RAM (VRAM) mapped into the CPU's address space. Accessing the VRAM is normally considered a safe operation. Some hardware accelerator registers are mapped to VRAM, but these can normally be excluded by the kernel code via the MMU of the host CPU.

From user-mode, the KGI driver API exposes a command interface that needs to do a user-to-kernel transition (under Linux, an ioctl call to /dev/graphic), and a memory-mapped linear framebuffer, a continuous area in RAM that represents the VRAM contents.

Not every graphics card has a linear framebuffer. However, as those of you who are familiar with DJGPP may know, there is an elegant solution for this: the MMU. If the card exports a banked-style buffer (for example, a 64K window at 0xA0000, as old Trident 8900s did), it is mapped at the appropriate place in a virtual memory area as big as a linear buffer of the card would be. The other areas are marked to be swapped out. If such an area gets hit, the driver is notified, moves the card's window accordingly, and corrects the mapping.

There are some speed problems with this, because the MMU trap is expensive compared to just setting the bank with an "out" instruction. At the same time, due to the design of most such cards, we cannot export the banking register to user space anyway, because of security considerations (it is normally on an indirect register that also hosts CRTC timing, and so on). On the other hand, this approach leaves bank-crossing-detection to the MMU and thus saves unnecessary (sometimes nontrivial) checking code.

Now, we have a decent and fairly fast interface for all common tasks. All really primitive things that are not worth the overhead to call into the kernel (DrawPixel, very short lines, and so on) are performed via the MMAPed VRAM. More complex and administrative functions are performed via the command interface.

One other catch is that you probably do not want to write any emulation code into the drivers for cards that do not have a particular function accelerated. Microsoft's DirectX handles this problem using capability bitmaps. Having capability bitmaps means that you can query to see if an acceleration function is available via some kind of a bitmap or test for a NULL pointer. In our opinion, this is too hard to extend, because you have to extend the bitmap or table with every new version, making lots of revision checks necessary to see if a particular capability is accessible in a given revision at all. So we chose another way to handle software fallback for our acceleration code.

An accelerated function call always returns a status code that either says "completed successfully" or an error code that suggests what to do instead and also how long that information is valid.

The suggestion can say:

- CANNOT: This is returned for hardware-specific operations or context-sensitive operations (for instance, trying to set the frame for video overlay on a board without such capability).
- USE_LOWER: This is used when it is most likely a good idea to use a set of simpler acceleration calls (for example, using multiple horizontal lines to draw a box), because the resulting calls would be accelerated.
- USE_MMAP: This is returned when no simpler accelerator calls are supported. Thus, it is advisable not even to try them, but rather to directly draw on the MMAPed VRAM.

The expire information tells how long this information is valid. This allows us to avoid having to call the accelerator function each time for cases where a certain accelerator function may be only temporarily unavailable.

- NOW: Retry next time. It can't be done just now, because the accelerator is too busy or some similar problem that is likely to go away the next time the function is called.
- GC: Retry when the graphics context has changed (for example, if the accelerator cannot draw with a given raster operation).
- MODE: Retry when the mode has changed (that is, if the accelerator cannot be enabled in a specific mode as in the VGA compatibility modes of many common accelerators).
- ALWAYS: The accelerator never has this capability.

The advantage of handling software fallback this way over a DirectX-style bitfield is that this is extensible in a compatible way on both kernel and user sides. A newer KGI driver will know some new command codes that older libraries won't know about. So, you could lose a bit of extra acceleration with older libraries, but it's better than being incompatible.

A newer library may use some command codes that are not supported by older drivers. This triggers a "default" case that deals with the commands and always returns ENOSUP_ALWAYS_LOWER or ENOSUP_ALWAYS_MMAP (depending on whether or not the driver has a reasonable base set of accelerated commands). This return code causes the library to permanently disable the accelerator call after the first try and use an emulation routine instead. Again, you may lose a bit of potential acceleration if your kernel isn't up to date with the library, but it still works.

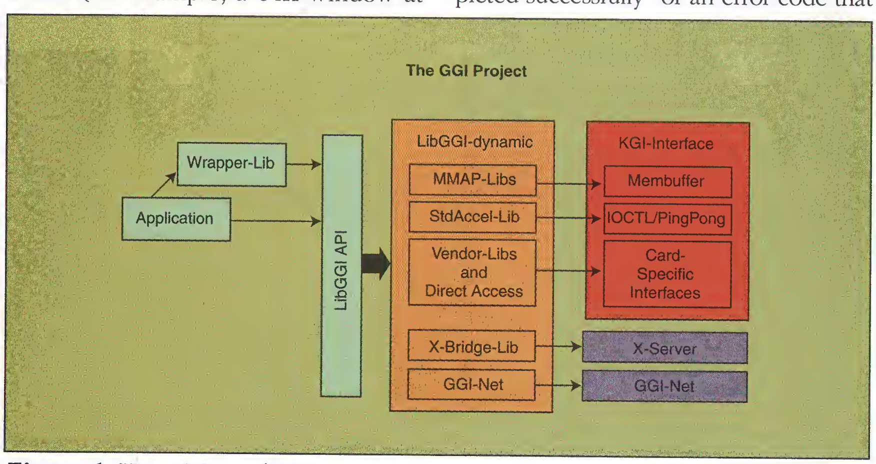
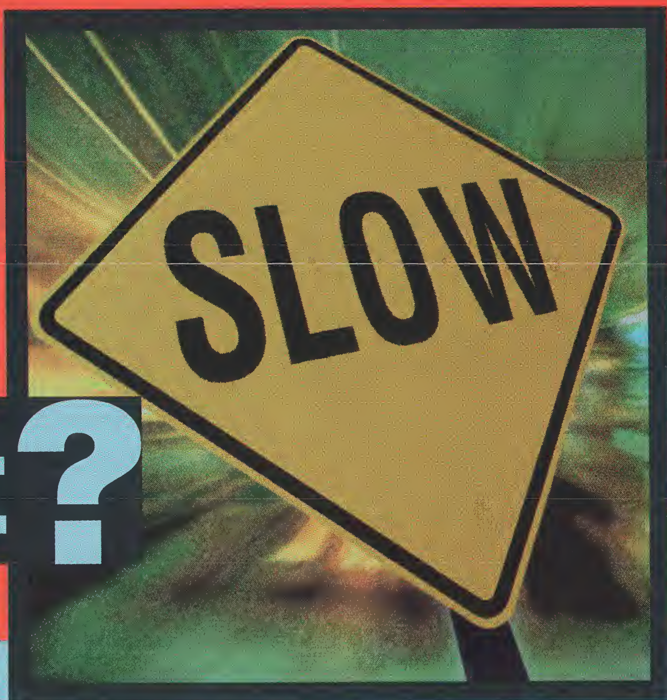


Figure 1: *How GGI and KGI are related.*

IT'S A CODE PERFORMANCE PROBLEM.

BUT WHERE?



ONLY NuMEGA TrueTime™ PROVIDES SEAMLESS MULTI-LANGUAGE PERFORMANCE ANALYSIS FOR COMPONENT-BASED APPLICATIONS.

Now, for the first time, you can easily and accurately pinpoint code performance bottlenecks no matter what language they are written in or where they reside. In Visual Basic, Visual C++, and Java, NuMega TrueTime lets you see exactly where problems occur. Get NuMega TrueTime and get all your applications and components up to speed.

Microsoft Enterprise Development Partner

Compuware® NuMega™

Call to request a FREE 14 DAY LICENSE.

1-888-686-3430 www.numega.com

SmartDebugging™ for faster development VISUAL BASIC® VISUAL C++® JAVA™

30-DAY MONEY BACK GUARANTEE

(continued from page 52)

Enhancements

While the scheme described earlier is enough for normal applications, there have always been some drawbacks to this approach:

- Only relatively common acceleration commands are supported. Adding all the card specifics would result in an incredible number of commands. In addition to the number of commands increasing astronomically, it is very possible that different accelerator functions in two different drivers could end up using the same command codes, as drivers are developed independently.
- There is no direct way to get at the acceleration registers, even if this is otherwise safe to do (which it is for a few very high-end cards).
- Some cards have multiple memory areas for textures, overlays, and so on.
- There is no way to support display lists or similar things that would dramatically reduce the number of user-to-kernel transitions and, thus, overhead.

To overcome these limitations, KGI allows exporting additional API functions that allow you to circumvent these problems:

- Private commands. KGI reserves an area for private command codes. These are handled by a card-specific library in user space to make the best possible use of the card.
- Mapping of card Memory-Mapped IO (MMIO) areas, or eventually allowing access to the card's ports if this is safe (up to now, we have not found cards where port access is safe). Here, too, card-specific libraries are used to convert the card-specific API represented by the MMIO area to the common API.
- Mapping of cards' additional memory areas like texture memory, YUV overlay planes, and so on.
- PingPong buffers, which are simply filled with commands (all in user space) and then executed with a single command (one user-to-kernel transition). This operation can be done asynchronously with the program continuing to execute on the host CPU, while the accelerator is fed with commands using either DMA, accelerator-generated "accel-idle" or "accel-buffer-lowwater" interrupts, or host-generated timer interrupts. This allows for maximum throughput, as the host CPU can prepare the next drawing commands while the accelerator is still drawing the last batch.

The EvStack Kernel Enhancement

EvStack is an extremely flexible console system we designed to overcome some limitations of the current Linux-KGI kernel patch, which breaks some features and programs (notably XFree and SVGAlib). The basic idea behind EvStack is to pass events between independent modules instead of hardwiring the calls. This allows you to plug together a console and dynamically swap out parts, like the VT-emulation. Under EvStack, you can have xterm, Linux, and dumb consoles on the same machine as well as different fonts, screen sizes, and screen modes (for instance, graphical consoles) on the different virtual terminals. With the EvStack patch installed, you can do one of three things:

- Turn EvStack off at compile time, giving you traditional Linux console code.
- Turn EvStack on, but load or compile in the conlinux.o module, giving you traditional Linux behavior running on the new code.
- Turn EvStack on, but don't load conlinux.o, giving you pure EvStack behavior, with the additional conlinux API disabled and all configuration occurring via */proc*.

—A.B.

Multiple APIs and Libraries

I have talked about having multiple APIs. How do you know which particular APIs are present and how to make use of them? How do you avoid a horrible mess where the applications must know all of the APIs?

This is one of the reasons for LibGGI, which consists of a basic stub library and a rather large bunch of API libraries that build the bridge between the various hardware (or software—LibGGI can also be used to display in an X-Window) APIs and the LibGGI API. When setting up a mode, LibGGI asks the target (KGI in our case) for a list of the exported APIs, a set of strings that classify how you can access various card features. Figure 2 shows a typical API list. The meanings of the strings, which are listed in increasing order of precedence; see Table 1.

```
"generic-linear-8"
"generic-ioctl"
"generic-ramdac"
"S3-generic"
"S3-virge"
```

Figure 2: *Typical API list.*

The libraries are loaded in a way that allows more specific functions to overload the more generic ones, automatically yielding a startup configuration that always uses the best available function. In some cases (as with the ioctl API), these entries can be altered at run time if functions are not available.

One problem remains. LibGGI can only make use of functions that are needed for implementing the LibGGI API. If you look at these functions, you will realize that they account for few of the functions a card can support.

We have decided to keep LibGGI small to save space for simple applications and things like embedded systems. For more complex functions, LibGGI allows the registration of extensions like LibGGI2d and Mesa-GGI, which add support for the APIs necessary for specific tasks.

Implementation Details

Additional goals with the design of KGI included:

- Easy driver writing.
- Modular design for cards that are made from similar components (S3 cards with different RAMDACs, clocks, and so on are a good example).
- A simple way to enhance drivers for fairly compatible future generations of known cards.
- Full abstraction from the operating system for easy portability.

These are achieved by using a modular design approach that makes every KGI driver consist of six basic modules:

- Chipset module. This controls all functions related to mode setup, CRTC programming, RAM timing characteristics, interfacing RAMDAC and Clock, and so on.
- Clock module. This controls the pixel clock generation. This is separated from the chipset driver, as there are cards (S3, for instance) that have the clock as a physically distinct chip, with the different cards made by combining basic chipset, clock, and RAMDAC chip in different ways.
- RAMDAC module. The RAMDAC modules is similar to the clock module, but controls the RAMDAC features like palette setting, VRAM-bus activation, RAMDAC-internal hardware cursors, Gamma correction, and the like.
- Graphics (accelerator) module. Some chips have the acceleration engine either detached from main chipset or use the same or very similar acceleration engine on different chipset versions. Thus, separating acceleration programming

from the other aspects of the card makes sense (that is, all newer S3 cards can be run with the S3 generic acceleration driver). Not all of the capabilities of very new cards would be used, but driver development is eased quite a bit, since you can try out your new chipset driver without having to write a graphic module.

- Monitor module. What features are there in a monitor that will need a driver? At the very least, such things as timing limitations, ensuring that the image is centered on the screen, power-saving capabilities, and more. Being able to use any of these requires some knowledge about what the monitor supports. The monitor driver allows safe access to these features, and automatically chooses suitable timings.
- Kernel module. This does the interfacing to the host OS. It implements access methods to the hardware, to PCI services, and so on. In theory, we should be able to run the same KGI driver on different operating systems by just linking with a different kernel module. (We have not yet tried this because we are currently restructuring the Linux console. Porting efforts now would result in a lot of duplicate work.)

Conclusion

What does Linux gain by using KGI? First, the graphics card is handled like any other device, which means that arbitration and access to critical registers occur in one central place—the kernel.

Second, since the kernel is able to control the graphics card, we have a few new capabilities:

- A real Secure Attention Key (SAK) that can kill off graphical applications safely because the kernel itself is able to reset the graphics card to a sane state.
- Simple and safe resizing capabilities for VTs. For example, with KGI, you can implement VT100 ESC codes that were impossible to implement without these resizing capabilities.
- Support for graphical consoles, thanks to the new EvStack kernel enhancement (see the accompanying text box entitled "The EvStack Kernel Enhancement"). This is immensely desirable for hardware that has no VGA-like text mode or for languages that require the ability to represent more than 256 characters.
- The ability to operate the graphics card in MMIO mode, which means that the registers of the card are mapped to a programmable place somewhere in the memory address space, thereby freeing the VGA registers in IO space. As a result, Linux/KGI is multihead capable with cards that support that feature.

String	Meaning
S3-virge	This is an S3 Virge card. If you have a specific library that knows the API and the Virge-specific functions, then load it.
S3-generic	The kernel knows which functions are available on all S3 cards.
generic-ramdac, generic-ioctl	The generic RAMDAC APIs are supported, as is the KGI-ioctl interface.
generic-linear-8	The card (for this mode, anyway) has a linear framebuffer with eight bits per pixel.

Table 1: *The meaning of the strings in Figure 2 (from bottom to top).*

Third, together with LibGGI, you have a lightweight, portable, and fast graphics subsystem. (A single-disk demo that uses a mere 700-KB compressed image is available electronically; see "Resource Center," page 3, or my home page at http://sunserver1.rz.uni-duesseldorf.de/~becka/.) This is of special interest for embedded systems, which can now use Linux instead of relatively expensive and less open ("nice README, but where is the source?") solutions like QNX or Windows CE.

Finally, you will no longer have dangerous SUID root graphics applications. The GGI project has developed both a wrapper library that allows most SVGAlib applications to run without root permissions, and a replacement X server called Xggi.

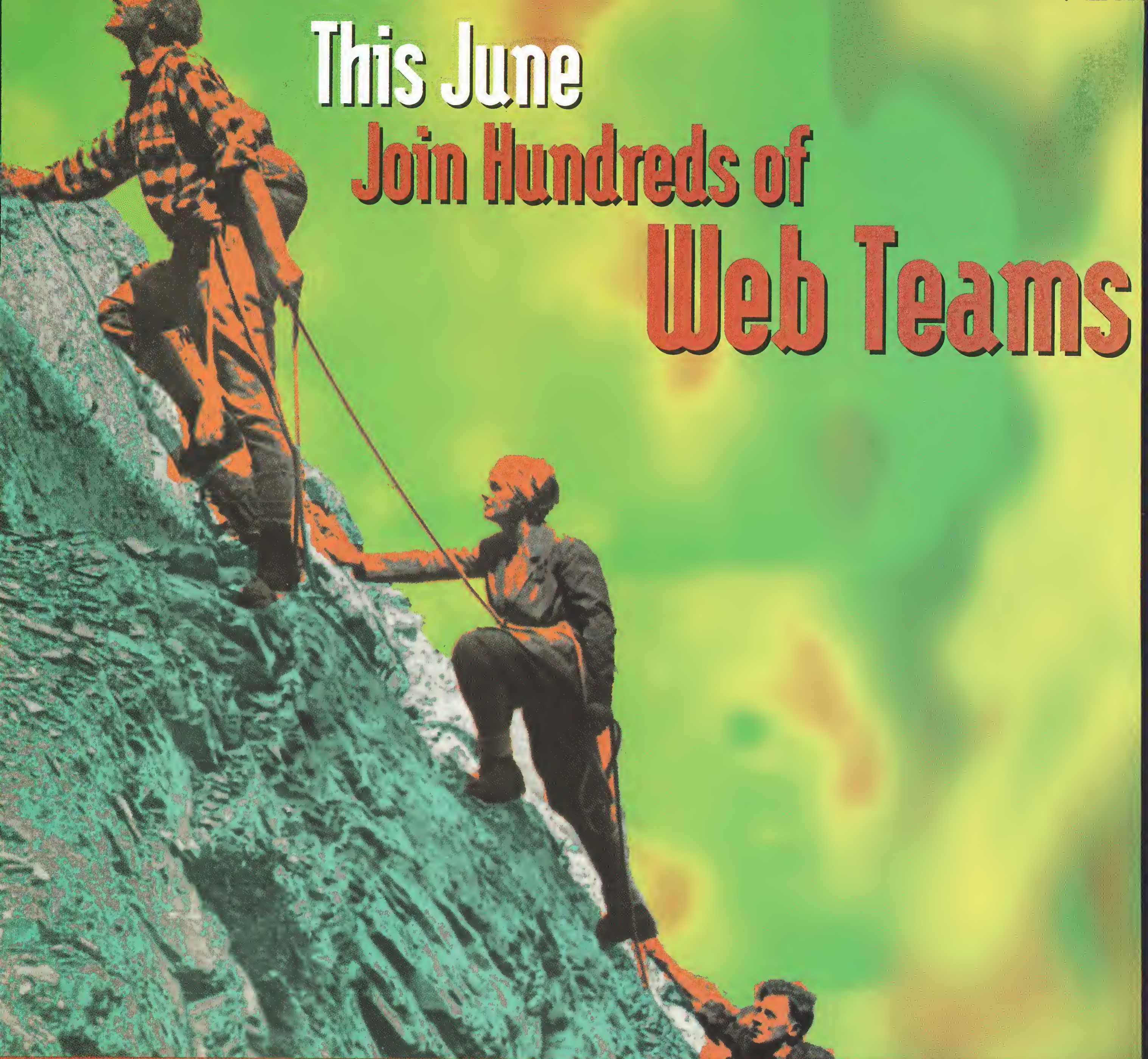
Resources

The GGI homepage (http://www.ggi-project.org/) contains snapshots of the latest source, instructions on how to obtain them via CVS, links to GGI-relevant web sites, and up-to-date information about the project. Our mailing list is hosted at ggi-develop@eskimo.com. Subscription information is found on the GGI web site. If you plan on subscribing, be prepared—the list has high traffic.

Acknowledgments

Thanks to the GGI development team, especially Steffen Seeger, Jason McMullan, Emmanuel Marty, Ben Kosse, and Michael Krause for their work and for reviewing this article and correcting several glitches. I'd also like to thank S3, Cyrix, 3Dlabs, the FLUG for providing the GGI development team with information and donations, and all the users and testers of GGI.

DDJ



What's New This Year

- ***home page*** sneak preview! Doug Block's social/cultural phenomenon comes to Web 98 for a sneak preview - world premiere will be on Cinemax later this summer. A lucky few will win tickets - don't miss it!
- World-class full-day **Tutorials** covering everything from Ecommerce (2 days in-depth) to Perl and Javascript.
- Mindbending **Open Sessions** and **Showcases**: Jakob Nielsen on Web 2003, Flash and Quicktime in all their splendor, and much more!
- Join the **WebScene** at **www.mfweb.com!** It's a distributed, user-run community, built around sharing web links and talking about them in a common forum. Discuss the conference topics and get the buzz going long before the conference even starts!
- **1998 Web Tools Awards** hosted by *Web Techniques* and Web Review.

Who's Speaking

Brian Behlendorf – *Backend*
Emily Davidow – *Strategy/Visual Design*
Abbe Don – *Information Design*
Dale Dougherty – *Strategy*
Bruce Ecke – *Programming*
Paul Haeberli – *Visual Design*
Kevin Lynch – *Visual Design*
Mark Meadows – *Visual Design*
Douglas Meisner – *Visual Design*
Peter Merholtz – *Information Design*
John Maeda – *Visual Design*
Drue Miller – *Information Design*

Make History. Register now for Web'98.

Miller Freeman

Conference: June 21-25, 1998 – Exhibition: June 22-24, 1998 – Moscone Center, San Francisco

Affine Texture Mapping

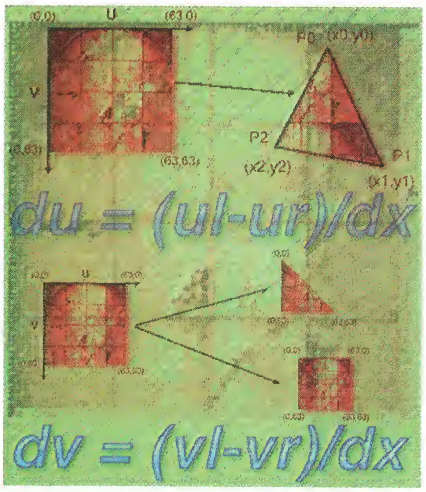
A fundamental technique for graphics programmers

André LaMothe

It used to be you could get away with developing flat shaded 3D computer games and engines. But if your software doesn't support texture mapping these days, it will likely end up in the bargain bin. One form of texture mapping is "affine" texture mapping, which is fundamental to many forms of 3D rendering, including light interpolation and other sampling type operations. In this article, I'll present an affine texture mapper that can texture map a 64×64-pixel/256-color rectangular bitmap onto a triangular polygon with full texture coordinate support. In addition, I'll include a demo that loads texture maps and draws thousands of textured triangles a second. Although this demo is in DirectX, the ideas and concepts are applicable to other systems. And since the texture mapper is in straight C, it's totally portable.

Getting Down to Specifics

Assume you want to texture map a rectangular bitmap that is 64×64 pixels in 256 colors (one byte per pixel) onto an arbitrary triangle with any coordinates. To do so, you need to take rotation and scaling of the triangle into consideration. To design the algorithm that makes this possible, I've labeled a number of points of interest on Figure 1. First, the destination triangle is made up of three vertices—*p0*, *p1*, and *p2*, with coordinates (*x0,y0*), (*x1,y1*), and (*x2,y2*), respectively. In addition, the axes around the texture map are U and V, where U is the horizontal



axis and V is the vertical axis. Both U and V range from (0,0) in the upper left to (63,63) in the lower right. What you want to do is design an algorithm that samples the texture map, so that the sampled pixels can be used to color each pixel of each scanline of the target triangle polygon as it is being rendered.

There are a number of ways to draw triangles, including tracing the edges of the triangle with a line-drawing algorithm (such as Bresenham's) or with simple interpolation. I prefer interpolation since it's more straightforward. Also, the concept of interpolation is important because the texture mapping algorithm is based on it. In Figure 2, all you have to do is find the points (shown as little dots) that make up the integer rasterized version of the triangle. Once you find these dots for each scanline that makes up the triangle, drawing the triangle is nothing more than performing a memory fill from dot to dot. Finding these points simply involves interpolating the slope of each side of the triangle. The interpolation is done as follows:

You know that the height of the triangle is:

```
dy=(y2-yØ);
```

and the difference in the "x" between the lower-left vertex and the lower-right vertex is:

```
dx_left_side=(x2-xØ);
dx_right_side=(x1-xØ);
```

Thus, the slope of the left side is:

```
slope_left_side=dy/dx_left_side
    =(y2-yØ)/(x2-xØ);
```

And, the slope of the right side is:

```
slope_right_side=dy/dx_right_
    side=(y2-yØ)/(x1-xØ);
```

However, you don't exactly want the slope. The slope is the "change in Y per change in X." This means that if you were to move over exactly one pixel in the X direction, then the Y would change by the slope. You don't want this. In fact, you want the opposite—*dx/dy*—because you are drawing the triangle scan line by scan line and incrementing Y each time; hence *dy=1*, which is a constant. Thus:

```
dx_left_side=1*(x2-xØ)/(y2-yØ);
```

and

```
dx_right_side=1*(x1-xØ)/(y2-yØ);
```

Listing One (listings begin on page 96) is a pseudocode implementation of the triangle drawing algorithm.

André is the author of Tricks of the Game Programming Gurus, Teach Yourself Game Programming in 21 Days, The Black Art of 3D Game Programming, *and his latest creation,* Windows Game Programming for Dummies. *He can be contacted at necron@slip.net or at http://xgames3d.com.*

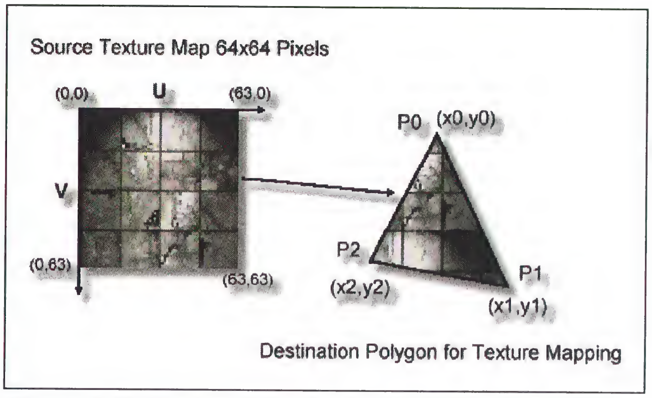


Figure 1: *Texture mapping source-to-destination labeling.*

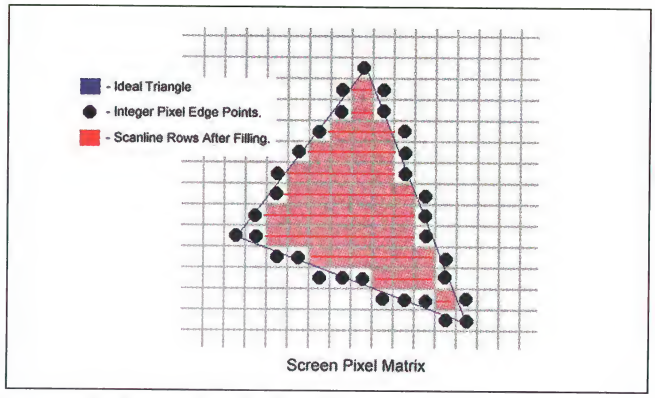


Figure 2: *Screen pixel matrix.*

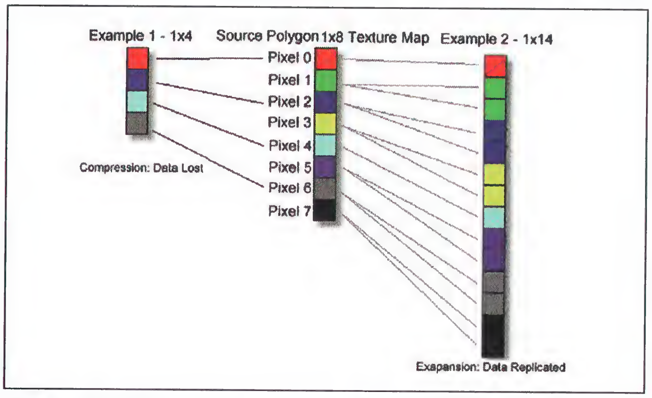


Figure 3: *1D texture mapping: (a) 1×4; (b) 1×14.*

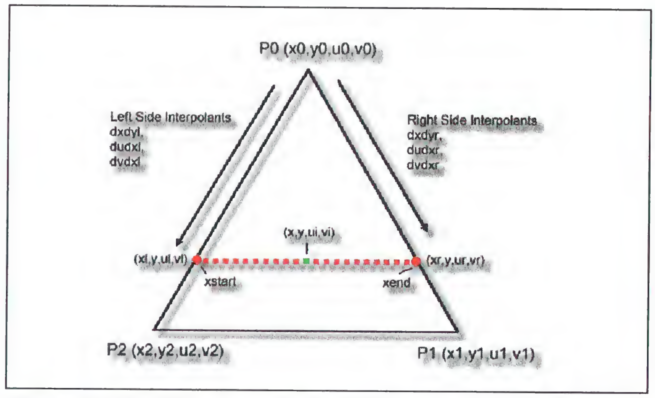


Figure 4: *Graphic representation of texture-mapping algorithm.*

One Dimensional Interpolations

Texture mapping a triangle with a rectangular texture map involves lots of interpolating. Consequently, it's easy to make a mistake or to write a slow algorithm. With this in mind, I'll start with the simplest case in one dimension. Figure 3 illustrates the simplest texture mapper—the texture mapping of a single vertical line that's one pixel thick and eight pixels high.

What you need to do is "sample" the texture map (in this case, a single 1×8 pixel bitmap) and map it into the destination polygon, which is 1×*n* pixels, where *n* can range from one to infinity.

As a first example, assume that your destination polygon is 1×4 pixels. It makes sense that you want to sample the source texture every other pixel, as in Figure 3. Thus, if you select pixels (0,2,4,6) of the source texture and map them into the destination polygon at positions (0,1,2,3), then you are doing pretty good. But how did you arrive at (0,2,4,6)? The answer is by using a sampling ratio, which is nothing more than an interpolation factor. In general, *sampling_ratio=source_height/destination_height*. Thus, the sampling ratio is *sampling_ratio*=8/4=2. Thus, every one pixel you move on the destination polygon in the vertical axis, you must move two pixels on the source to keep up. That's where the "two" comes from and hence the sampling sequence (0,2,4,6). Unfortunately, this means you had to throw away half the pixels. This is a problem with sampling on an integer matrix without any averaging. If you were writing a high-end 3D modeler (like 3D Studio MAX), then you would probably average the pixels you're sampling (area sampling) to get a better approximation, but for games and real time, our technique will do.

In the previous example, the source texture was compressed; that is, the destination was smaller than the source and information was lost. On the other hand, there could be the case that the destination is bigger than the source, and there isn't enough information to go around. In this case, the source data must be sampled more than once and replicated. This is where all "chunkiness" comes from when texture mapped polygons get too close to you in a 3D game. There isn't enough texture data so some sample points are sampled many times, creating big blocks. Referring again to the second example in Figure 3, you see that the source is again 1×8, but this time the destination is 1×14 pixels. Obviously, you need a fractional sampling ratio. Again, *sampling_ratio=source_height/destination_height;*. Thus, the sampling ratio is *sampling_ratio=8/14=0.57.*

Hence, the sample for every pixel you draw on the destination polygon should be taken 0.57 units from the last sample point on the source. This gives you the following sample point sequence for destination pixels (0,1,2,3,....13):

Sample 0: 0.57
Sample 1: 1.14
Sample 2: 1.71
Sample 3: 2.28
Sample 4: 2.85
Sample 5: 3.42
Sample 6: 3.99
Sample 7: 4.56
Sample 8: 5.13
Sample 9: 5.7
Sample 10: 6.27
Sample 11: 6.84
Sample 12: 7.41
Sample 13: 7.98

To get the actual sample points, you simply truncate the sample points in integer space or take the floor of each value resulting in the sample points (0,1,1,2,2,3,3,4,5,5,6,6,7,7), which sounds about right. Each point got sampled about two times, or 1/0.57.

Multiple Interpolations

When I wrote my first affine texture mapper, I thought something must be wrong since it seemed like I was interpolating everything. The truth is, there is really no way around all the various interpolants, and in the end, the inner loop

for each pixel can be optimized into around 10 cycles/pixel on a Pentium, which translates to a theoretical maximum of 10- to 20-million textels (textured pixels) per second on a 100-MHz Pentium.

The idea behind the algorithm is that you want to interpolate down the left and right edges of the triangle and draw each scanline strip as we go with the proper texture pixels. What you need to do first is assign full texture coordinates to the vertexes of the destination triangle to give us a frame of reference for the interpolants. Thus you must assign each vertex a (u,v) texture coordinate, as in Figure 4. Therefore, each vertex has a total of four data components—that is, it's a 4D value. Since the source texture map is 64×64 pixels, the texture coordinates must range from 0–63 for any vertex. This will map or stretch the texture map to each vertex.

Figure 5(a), for example, has the texture coordinates (0,0), (63,0), and (63,63) mapped to vertices 0,1, and 2, respectively. This basically copies half of the texture map to the destination triangle, which is what you would expect. In Figure 5(b), you see the same texture mapped onto two triangles which are adjacent to each other forming a square. In this case, the texture coordinates are selected in such a way that half of the texture map is mapped to one triangle and the rest to the other, hence, a perfect texture wrapping around two triangles. Moreover, this is how you would make a quadrilateral; that is, with two triangles. Now that you have a visual on the problem and know the labeling from Figure 4, let's implement the algorithm mathematically. The variable names used in the following analysis are based on Figure 4 and the final program so that you can follow the program code more easily.

The left edge interpolants are:

```
dxdyl    =    (x2-xØ)/(y2-yØ);
  // x interpolant for left side
dudyl    =    (u2-uØ)/(y2-yØ);
  // u interpolant for left side
```

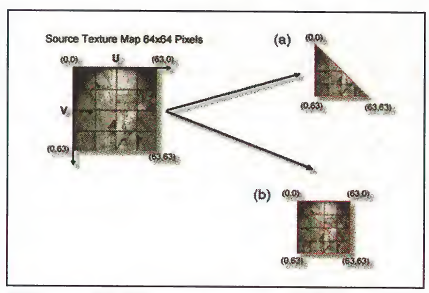


Figure 5: Texture mapping 3- and 4-sided polygons. (a) single triangle; (b) two triangles making a quadrilateral.

```
dvdyl    =    (v2-v0)/(y2-y0);
  // v interpolant for left side
```

Similarly, the right edge interpolants are:

```
dxdyr    =    (x1-x0)/(y2-y0);
  // x interpolant for right side
dudyr    =    (u1-u0)/(y2-y0);
  // u interpolant for right side
dvdyr    =    (v1-v0)/(y2-y0);
  // v interpolant for right side
```

There's a lot of room for optimization. For example, $(y2-y0)$ is common and need only be computed once. Furthermore, it's better to compute the reciprocal of $(y2-y0)$ and then multiply.

The interpolants must be in reference to some starting point. This starting is the top-most vertex, vertex 0. Hence, you need to start the algorithm off in the following manner:

```
xl = x0; // starting point for
  left side edge x interpolation
ul = u0; // starting point for
  left side edge u interpolation
vl = v0; // starting point for
  left side edge v interpolation
```

And for the right side,

```
xr = x0; // starting point for
  right side edge x interpolation
ur = u0; // starting point for
  right side edge u interpolation
vr = v0; // starting point for
  right side edge v interpolation
```

Now you can interpolate down the left and right edges with:

```
xl+=dxdyl;
ul+=dudyl;
vl+=dvdyl;
```

and

```
xr+=dxdyr;
ur+=dudyr;
vr+=dvdyr;
```

At each point on the left and right edge of the triangle, you still need to perform one more linear interpolation across the scanline. This is the final interpolation and the one that will give you the texture coordinates *(ui,vi)*, which you'll use as *[row, column]* indexes into the texture bitmap to obtain the textel. All you need to do is compute the *u,v* coordinate on the left and right side, then use the *dx* to compute a linear interpolation factor for each. Here's the math:

```
dx = (xend-xstart);
  // difference or delta dx
xstart = xl;
  // left starting point
xend = xr;
  // right starting point
```

Therefore, the interpolants across each scanline in *u,v* space are:

```
du = (ul-ur)/dx;
dv = (vl-vr)/dx;
```

Then with *du,dv*, you have everything you need to interpolate across the scanline at vertical position y from *xstart* to *xend*; see Listing Two.

Conclusion

That's it. Of course for the outer loop, you would still interpolate *xl,ul,vl,xr,ur,vr* down the triangle edges for each scanline of the triangle.

The files tmapper.h and tmapper.cpp (available electronically; see "Resource Center," page 3) provide a complete implementation of the texture mapper. The program assumes a specific input data structure and that the texture map is a linear bitmap 64×64 pixels. Other than that, it's nothing more than an implementation of the derivation here, along with all the triangle cases and clipping. In addition, the program tmapdemo.cpp (available electronically) is a complete DirectX demo of the texture mapper that draws random triangles all over the screen in 640×480×256. Finally, BOX2.EXE is a 3D demo written by Jarrod Davis that uses the texture mapper.

DDJ

(Listings begin on page 96.)

Inside DVD

More storage, more features, same size

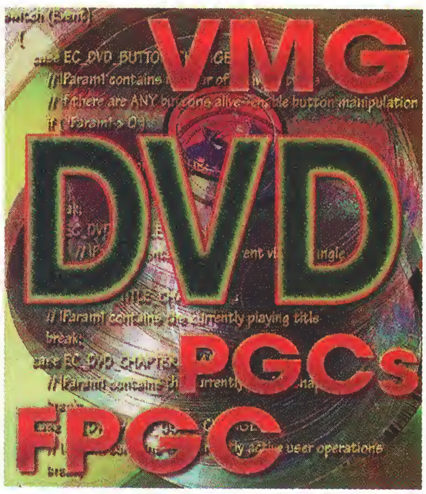
Linden deCarmo

Although DVDs physically resemble CD-ROMs (five inches in diameter and 1.2 mm in thickness), DVD stores between seven and 25 times more data. This huge storage capacity makes it an ideal distribution vehicle for full-length movies (up to four hours long), high-quality audio (the contents of up to 13 CDs can be stored on one dual-layer DVD), and similar applications (not to mention data storage). DVD has garnered support from all major electronics and computer companies, and many major movie and music studios.

Even though DVD technology is promising, technical details about it are scarce — in part because those details are still being worked out. Currently, specifications have been agreed upon for DVD-Video and DVD-ROM. As its name suggests, DVD-Video is for video programs and is played in DVD players connected to TVs. DVD-ROM, on the other hand, stores computer data and is read by DVD-ROM drives connected to computers. Variations on DVD-ROM include those that are recordable one time (DVD-R) or many times (DVD-RAM). Most computers with DVD-ROM drives can also play DVD-Videos. Finally, there's the DVD-Audio format, for which technical specs haven't yet been finalized.

In this article, I'll examine how a DVD-Video (or simply DVD) player operates, examine the features of a DVD title, and investigate the interactive capabilities in both computer and consumer DVD titles. (Also included with this article is a barebones, command-line DVD player, which is available electronically; see "Resource Center," page 3.)

DVD was conceived by the DVD Forum, a consortium of companies that includes Hitachi, JVC, Matsushita, Mitsubishi,



Philips, Pioneer, Sony, Thomson, Time Warner, and Toshiba. Although no one "owns" DVD, companies making DVD products must license patented technology from a pool of companies.

One result of this collaboration is the multivolume series *DVD 1.0 Specification for Read-Only Disc* (ordering instructions are available at http://www.mpeg.org/MPEG/DVD/General/Order.html). The most interesting book in this series is *Volume Three*, which focuses on DVD-Video — a combination of a reference player design, optical media format, and multimedia data structures. The DVD-Video specification describes the required features to which a hardware-independent virtual machine must adhere. It also defines the assembly-language opcodes that have to be interpreted, the state diagrams the player must enforce, the system registers that can be manipulated, and the size and capabilities of user-accessible memory.

The DVD-Video Specification

The DVD-Video virtual machine contains a low-level, assembly-like, instruction set with the usual branch, compare, and set operations found in most processors. However, the specification also has unique opcodes specifically designed for interactive presentations. For example, there are instructions to monitor parental controls, jump to specific locations in a presentation, and dynamically switch audio and video tracks.

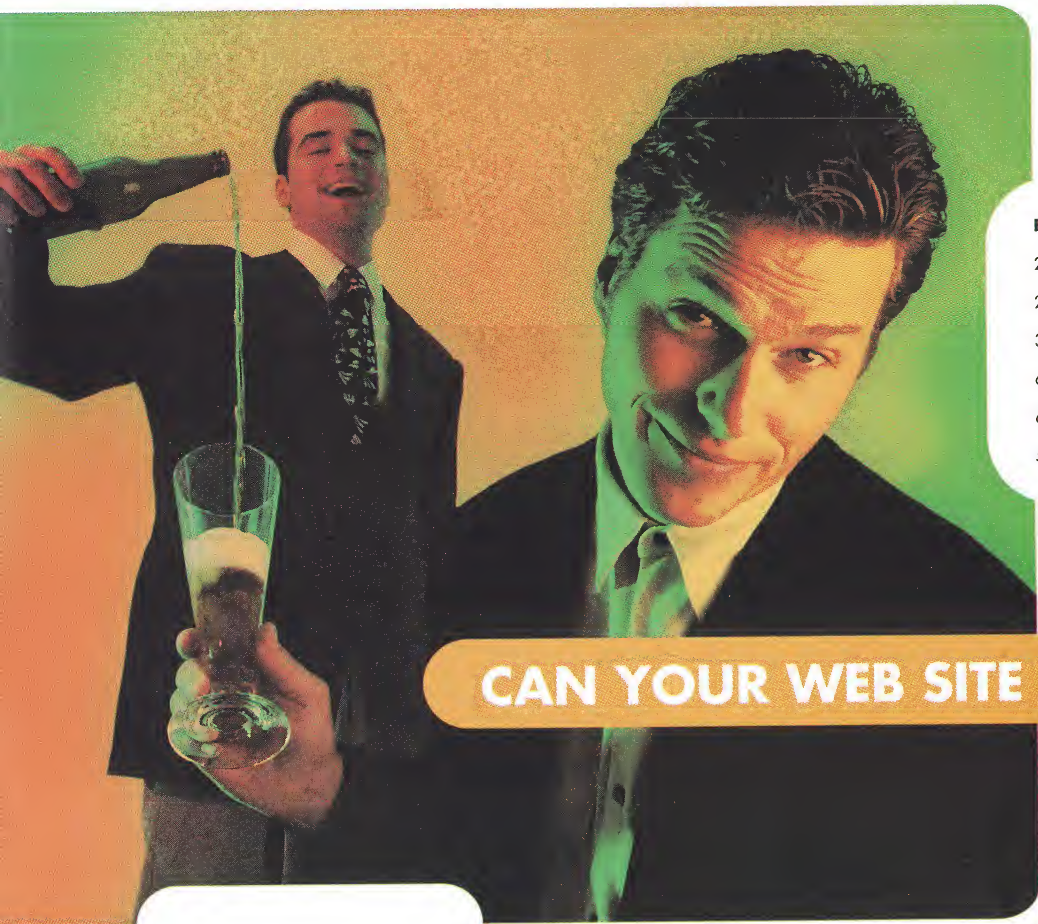
All DVD players have at least 20 system parameters (or registers) that can be accessed only by privileged opcodes. For instance, there are instructions to change the currently playing audio stream and update the system register, which monitors the currently playing audio stream number. The player also offers 16 general-purpose parameters that you can modify without special instructions.

Every DVD-Video disc contains a *video_ts* (or video title set) directory, which consists of files with IFO or VOB extensions. VOB files store multimedia data, whereas IFO files instruct the player how to play the content in VOB files. There are two types of IFO files — Video Manager (VMG) and Video Titles Set (VTS).

DVD Video Manager

VMG is found in the video_ts.ifo file and it is the first file all DVD players read. This file is similar to a boot sector on a floppy disk — it supplies the player with initialization information and then points

Linden is a software engineer at Oak Technology where he is currently working on the Interactive DVD Browser, the first publicly available DirectShow DVD environment. You can contact him at lindend@ibm.net.



BEERFINDER PROFILE

24 database tables

23 interactive database pages

345 page objects

60 interactive buttons

60 JavaScript functions

500 SQL functions

CAN YOUR WEB SITE DO THIS?

Dynamically publish your relational databases ?

Do it in a code-free way ?

While celebrating the foamy goodness of beer ?

Centura**net.db** helps just about anyone put a database on the Web – and we mean anyone. Witness Beerfinder, the Internet database devoted to listing all good brews, created by two wags in our Marketing department.

What began as a lark over some afternoon pints now has a permanent place on the Centura Web site. What better way, those clever Marketing guys Max and Dave thought, to demonstate that Centura**net.db** really lets you (or somebody without SQL or HTML programming skills, like Max and Dave) deploy real-time data from SQL databases to the Web?

Of course, your online data could be anything, even something more important than beer – like your company's product catalog. Centura**net.db** lets you navigate and edit live databases from your browser, right out of the box. It's ideal for creating a truly interactive Web site — self-help, self-service.

In fact, whatever you want a database to do, Centura can help. If two lushes in Marketing can get their database online, just imagine what you might accomplish with our tools.

GET A FREE TRIAL OF CENTURAnet.db NOW AT www.centurasoft.com.
(And to find out more about Beerfinder, go to www.beerfinder.com. Cheers.)

Centura **net.db**

YOU CAN DO IT WITH US.

centura™
—GUPTA

(continued from page 64)
the player to where it can obtain the remainder of the data necessary to continue playback.

VMG contains vital information such as the name of the disc, number of titles on the disc, and optional information such as parental controls (parental controls prevents little junior from looking at violent or explicit material). Although DVD defines eight numerical levels of parental protection, the name associated with each level may differ between countries (for instance, the Canadian rating system differs from the United States system). In all cases, higher parental level values always permit more content to be viewed (see the accompanying text box "Parental Levels"). For instance, Parental Level Seven ("NC-17" in the United States) enables you to see more movies than Level One ("G" in the U.S.).

A VMG may also contain a feature known as the "VMG Menu" (VMGM), which gives users an overview of the disc's contents and potentially lets users jump to specific points in the title. It is composed of a video stream and an optional audio stream and a subpicture stream. To avoid bugs in the first generation of players, most early menus used MPEG-2 still images and had minimal interactivity. Because newer players are more stable, innovative authors are including full-motion video and surround sound in their menus.

Interactive features in menus (such as background audio and video) are displayed and controlled by data structures called "Program Chains" (PGCs)— arrays of programs, each of which normally represents a screen within a menu or chapter in a movie title. Each program contains one or more cells. Cells let you divide menus or chapters into more granular or logical subdivisions. They last a finite period of time, may have command instruction (or DVD assembly opcode) associated with them, and can enforce a delay when they complete playback. Although few titles take advantage of multiple cells per program, the DVD specification enables this feature to support effects such as slide shows in which playback must pause for a specific period of time after displaying a cell (or image).

Besides programs, PGCs may contain up to 256 navigational commands (128 of which may be executed before the programs in the PGC are presented, and 128 thereafter). These navigational commands are used for interactive purposes such as modifying the current video angle.

Every PGC also contains User OPerations (UOPs), which are stored in 32-bit fields where each bit (or individual UOP) represents the status of a unique interactive function on the player. Because these UOPs dictate which features in a PGC are legal (or usable), they have been almost as controversial as the region codes (see the accompanying text box entitled "Region Management"). To illustrate why this feature is so contentious, examine the Fast Forward UOP bit. If this bit is set, the DVD player cannot fast forward for the duration of that PGC. As a result, tricky content creators can embed commercials in DVD content and users will not be able to fast forward past them!

Attached to the tail of the VMG is the First Play PGC (FPGC). Once the DVD player is initialized, it searches for this PGC and executes the navigational commands inside of it. Most titles contain FPGCs that cause the player to display the VMGM, although it is possible for the

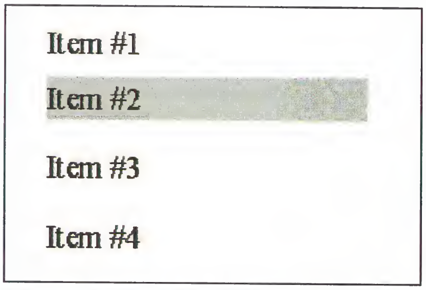


Figure 1: *How highlights work. The DVD player manipulates the color and contrast of a rectangle within the subpicture and this causes the area to appear highlighted. Here, the contrast for Item #2 is emphasized so that it appears selected.*

(continued from page 66)
FPGC to bypass the menu and jump directly into a movie scene (this is the technique that movies such as *The Mask* use to initiate immediate playback of a title).

Besides the video_ts.ifo file, the VMG also contains a Video OBject (VOB) file named video_ts.vob. VOB files are divided into packs each of which may contain a different media stream (packs are similar to a chunk in a WAV file). Although a pack may contain any data type, the DVD specification has stringent definition for video, audio, and subpicture packs.

Video packs in a VOB file normally contain MPEG-2 video. Although the MPEG-2 video format was defined by a standards body, it supports myriads of options that make it difficult to create a robust decoder. Therefore, to enhance compatibility and reliability, DVD places the restrictions of limited choice of resolution and maximum bitrate guidelines on MPEG-2 video content.

For NTSC locales (North America and Japan), the MPEG-2 video stream resolution in DVD must be 720×480, 704×480, 352×480, or 352×240. PAL (or European) resolutions must be 720×576, 704×576, 352×576, or 352×288. Furthermore, whatever video resolution and audio compression routines are used, the content cannot exceed a sustained bit-rate greater than 10.08 Mbits/sec.

The designers of DVD also delineated what audio packs may appear in a DVD stream. The audio types supported in the initial DVD specification include: Pulse Code Modulation (PCM), Dolby Digital (AC-3), MPEG-2 audio, Digital Theater Sound (DTS), and Sony Dynamic Digital Sound (SDDS). PCM is commonly used in stereo sound tracks and is identical to PCM content found in Windows, UNIX, and Macintosh (although DVD supports higher PCM resolutions and sampling rates than these environments).

If the content contains multichannel sound, then for all practical purposes, it contains AC-3 packs. To explain, the DVD specification states that AC-3 is mandatory for multichannel audio content in North America. By contrast, European (or region two) content initially mandated that MPEG-2 audio be the default multichannel audio standard. Recently, the Region Two Specification was modified to require either AC-3 or MPEG-2 audio for multichannel content. Since every other region in the world requires AC-3, it is likely that AC-3 will become the dominant format in Europe also.

Besides video and audio, VOB files also support subpicture packs. In DVD terminology, a subpicture is a Run Length-compressed bitmap. Each bitmap has a palette of 16 colors, four of which can be active at once. Up to 32 subpicture streams can exist in a given VOB file (usually one stream per language). Unfortunately, since the subpicture palette is so limited, it is difficult to create realistic effects with subpicture alone. As a result, many vendors combine subpicture with high-resolution MPEG-2 video.

The most noticeable use of subpicture is for closed-caption text. Behind the scenes, DVD also uses subpicture in menus. When a menu is displayed, the DVD player modifies the color and contrast of the subpicture for a particular area in the menu, the location appears to be highlighted or selected. As users traverse the menu, the subpicture rectangle is changed so that a selected area moves with them; see Figure 1.

Unlike conventional bitmaps, subpicture data in the stream may be attached to display instructions (or opcodes) that manipulate the image. For instance, there are opcodes that cause the subpicture bitmap to fade or scroll. However, the most interesting opcode is *forcedly start display*. Users often turn off the decoding of a subpicture so that they don't have to view foreign subtitles. When the DVD player encounters forcedly display opcode, the subpicture must always be decoded regardless of user preferences (this is why subpictures in menus will always be displayed even if subpicture decoding is turned off).

Woven among the subpicture, audio, and video are highlight packs. These

highlight packs contain user interface elements called buttons. Buttons are rectangular areas on the screen that monitor user input, and up to 36 buttons can be displayed at any single time. Each button contains associated highlight data structures, and these structures inform the player how to color a button when it is not selected, when it is selected, and when it is chosen (or activated). It also informs the player how long the buttons should remain on the screen and which numerical sequence on the player's remote control can select the button.

DVD Video Title Manager

Besides the VMG, every DVD Video disc contains one or more titles (or movies). These titles are stored in logical containers called "Video Title Sets" (VTS). Like the VMG, there is a strict naming convention for files in a VTS. All files in a VTS are in the form *vts_xx_y* where *xx* is the VTS number (up to a maximum of 99) and *y* is the index within the VTS.

Each VTS has a unique IFO file, vts_xx_y.ifo, and it uses the same data structures as the VMGM: PGCs, programs, and cells. Unlike VMG data structures, VTS data structures often use the exotic capabilities found in PGCs. For instance, title cells can have up to nine different video

Parental Levels

The initial wave of DVD titles had no parental enforcement. The second generation of titles (such as Disney and Universal) have parental controls, but are buggy. For example, Disney content requires players to be at Parental Level Eight (see Examples 1 and 2) before playback can commence, but Level Eight isn't even defined for the United States! Hopefully, as the content matures, parental enforcement will be less problematic.

—L.D.

```
Mov GPRMØ, SPRM13       ; get value of system parental register and copy
                        ;     into a user register #Ø
LT  GPRMØ, 8            ; if  parental level < Max parental level
                        ;     (i.e. 8)
GOTO    Failure         ; then alert user about the failure
```

***Example 1:** Poor parental checks in DVD content. Instead of checking for the parental level required by the disc, the content forces the parental level to be at least Level Eight before it will run.*

```
Mov GPRMØ, SPRM13        ; get value of system parental register and copy
                         ;     into a user register #Ø
LT  GPRMØ, #DISC_LEVEL   ; if parental level < the required parental level
                         ;     on the disc
GOTO    Failure          ; then alert user about the failure
```

***Example 2:** Correct parental checks in DVD content. In this case, the content verifies that the parental level in the player meets the minimum requirement for the disc, rather than the arbitrary Level Eight in Example 1.*

angles (an angle normally is an alternate camera angle of the presentation). Users can dynamically switch the viewing angle and the player will smoothly transition to a different location in the VOB file based on instructions in the cell.

Title PGCs can also utilize either simple or complex parental controls. Simple parental controls prevent a PGC from playing if users have not authorized its presentation. They will present users with a warning message indicating that the presentation cannot continue until users modify the parental setting for the player.

More sophisticated titles can dynamically select a different PGC depending on the current parental level. They have blocks of PGCs, only one of which will be displayed based on the parental setting. For example, if the player is set for G-rated movies, a nonviolent PGC will be chosen. By contrast, if the player has an R setting, an alternate PGC in the parental block with violent content will be displayed.

Unlike the VMG, a VTS categorizes menus into different topics: chapter, audio, angles, subpicture, and overall title control. These menus contain the same functionality as the VMGM including motion video, background audio, and interactivity via buttons.

Although interactivity has been the most hyped feature in DVD menus, they offer other intriguing options. For instance, DVD menus can be multilingual. To explain, when you create a logical menu screen, it can contain multiple versions of the menu, each in a different language. When the menu is displayed, the DVD player will check the current language system and pick the appropriate menu system for that language. Consequently, you can ship the same disc to different areas of the world, and the DVD players in each region will use the appropriate menu for that language.

Besides internationalization, DVD menus also support the same parental locking features found in title PGCs. You can use these parental controls to display completely different menus depending on the current parental rating system. For example, if the player's parental setting only permits G-rated movies, then the parental block would not show the default PG-13-rated menu, but instead show a special G-rated version that does not give viewers access to the chapters in the movie with sensitive content.

(Many early DVD developers wanted their titles to play on both Windows 95 and dedicated DVD Video machines. Because their programs used the Media Control Interface [which only uses VOB files and ignores the IFO required by DVD-Video], they had to create IFO files for DVD-Video compatibility. Unfortunately, they failed to follow the naming conventions for these files and were bitterly disappointed when they discovered that the content was unusable for DVD Video.)

The DVD specification also defines the minimum set of interactive functions (or operations) a player must provide to the user. Since these capabilities are found in Annex J of the specification, they are often referred to as "Annex J functions." These commands can be divided into the following categories: user interaction via

Region Management

Many users have manually modified their DVD player to ignore region management. This process is usually accomplished by setting all the bits (or enabling all regions) in the player's internal region register. This hack initially allowed players to play content from any region of the world. However, content creators have embedded instructions in the DVD discs to validate that the region code of the player is legitimate. If the content detects that the player has an invalid region code, it will refuse to play the movie (see Example 3).

VMG is also the source of a controversial field—region control. To explain, the DVD specification places artificial limitations on where the disc may be played. If the region code for the player does not match the region code in the content, then the DVD specification will not let the player present the disc—even though the player can decode the content.

—L.D.

```
Mov GPRMØ, SPRM2Ø       ; get the region code of the player
NE  GPRMØ, 1            ; If region code is not exactly ONE
GOTO     Failure        ; then either this is the wrong player or
                        ;      the user hacked it.
```

Example 3: *Region code checks in content. This sample verifies that the DVD player running the content can play Region One—and only Region One—content by ensuring that only one bit in the region control register is set. If multiple regions are enabled, it will fail.*

buttons, stream controls, random access to presentations, and menu manipulation.

The specification offers commands to navigate through buttons (*UpperButtonSelect()*, *LowerButtonSelect()*, *LeftButtonSelect()*, and *RightButtonSelect()*). Once you've decided on a button, you can use *ButtonSelect()* or *ButtonActivate()* to make a selection.

While a title is playing, you can change the viewing angle via the *Angle_Change()* method. *Audio_Stream_Change()* and *Subpicture_Stream_Change()* let you change audio and subpicture streams (or languages). If you're in a still condition (such as a pause between slides in a slide show), *Still_Off()* causes normal playback to resume.

There are a number of methods that enable random access to content. If you wish to search through the title, you can search via time (*Time_Search()* or *Time_Play()*), by chapter (*Chapter_Play()* and *Chapter_Search()*), or by title (*Title_Play()*).

The *MenuCall()* function lets you display a menu. It has one parameter that dictates which type of menu is displayed (Chapter, Audio, Subpicture, or Title). There are also methods to select Subpicture or Audio streams (*Subpicture_Stream_Change()* and *Audio_Stream_Change()*, respectively), modify parental settings (*Parental_Level_Select()*), and change angles (*Angle_Change()*).

Although Annex J defines the minimum set of interactive functions a DVD player must provide, it is legal, and in some cases, necessary to provide additional functionality for a specific platform. For instance, Microsoft's DirectShow for Win32 (a standard interface and the software drivers required for writing Windows-based DVD applications; see http://www.microsoft.com/directx/) provides enhancements that are specific to the computer environment and not addressed in the specification (see Listings One and Two; listings begin on page 96). It provides methods to process mouse input, finer control of the presentation, and support for asynchronous DVD events. Listing Two illustrates how you process DVD-related DirectShow events.

Conclusion

Unlike VHS, DVD is not simply a linear medium. It was designed to unite computer and consumer electronics users by offering high-quality video, multichannel audio, interactive functions, and a format that can adapt to future technologies. Furthermore, the DVD specification is hardware independent, so your content can run on a wide variety of devices. Once you begin to develop with DVD, you'll never again want to return to today's space constrained, postage-stamp-size multimedia world.

For More Information

Robert's DVD Info:
http://www.unik.no/~robert/hifi/dvd/

Kilroy's DVD FAQs:
http://www.CD-info.com/CDIC/Technology/DVD/dvd-faq.html

Chad Fogg's Technical Notes:
http://www.mpeg.org/~tristan/MPEG/DVD/

DVD-Video Production Guidebook:
http://www.nbdig.com/html/dvdmain.htm

Quantel Digital Fact Book:
http://www.quantel.com/dfb/

Sonic DVD Primer:
http://www.sonic.com/html/dvd/PDF/primer.pdf

DDJ

(Listings begin on page 96.)

68HC05-Based Peripheral Devices: Part II

The keyboard interface as a power supply and communications link

Derrick B. Forte
and Hai T. Nguyen

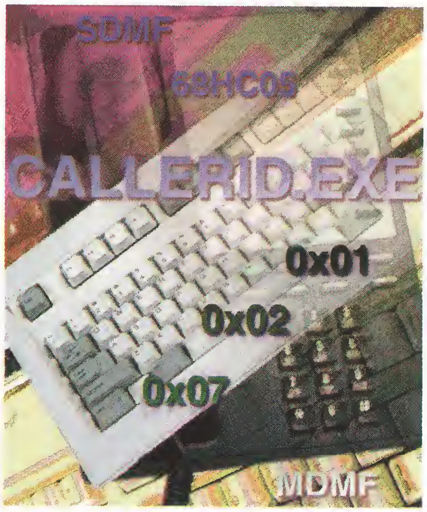
Although the IBM AT's keyboard port is the primary means of user input, you can also use it as the power supply and communications link for small low-power computer peripherals. In this two-part article, we present a system model around which such peripherals can be designed. The application we present is a Caller ID peripheral device based on the Motorola MC68HC(7)05P9 microcontroller. This device is capable of receiving Caller ID transmissions and displaying the received data on an AT-compatible computer. Last month, we focused on the Caller ID protocol and hardware design issues. This month, we complete our discussion of the hardware and zero in on the software.

The authors are engineers at Motorola, and can be contacted at r20367@email.sps.mot.com.

The Caller ID Data-Acquisition Block

The Caller ID data-acquisition block performs two functions within the application's system design:

- Provides an electrical interface to the telephone line.
- Demodulates and validates the Caller ID analog signal and converts it to a digital bit stream.



Though many Caller ID designs implement these functions with discrete analog circuitry, we selected a more integrated solution for this application—Motorola's MC145447 Calling Line Identification Receiver with Ring Detector. This device provides the needed interface to the telephone line, demodulating the BFSK asynchronous data signal, and outputting a digital stream. The design of this block was largely taken from the application note section of the technical data sheet for the MC145447. The device also has a number of signal validation and power saving features that are useful for Caller ID designs for which low power-consumption is an issue. Since this application is powered by the host computer's keyboard interface, it does not use any of the MC145447's power saving modes.

The MC145447's interface to the telephone line's twisted pair can be divided into two types of signals: Caller ID data acquisition signals and ring detection and validation signals. The ring detection and validation signals serve to detect the presence of a valid ring signal on the twisted pair and participate in bringing the device out of power-down mode. There are four signals that comprise the ring detection and validation portion of the interface. Three of the signals—Ring Detect IN 1 (RDI1), Ring Detect IN 2 (RDI2), and /Ring Time (/RT)—are inputs. There is also one output—/Ring Detect Out (/RDO)—which is asserted when a valid power ring is detected on the telephone line twisted pair. The /RT pin works in conjunction with the RDI1 pin to generate internal signals that are part of the

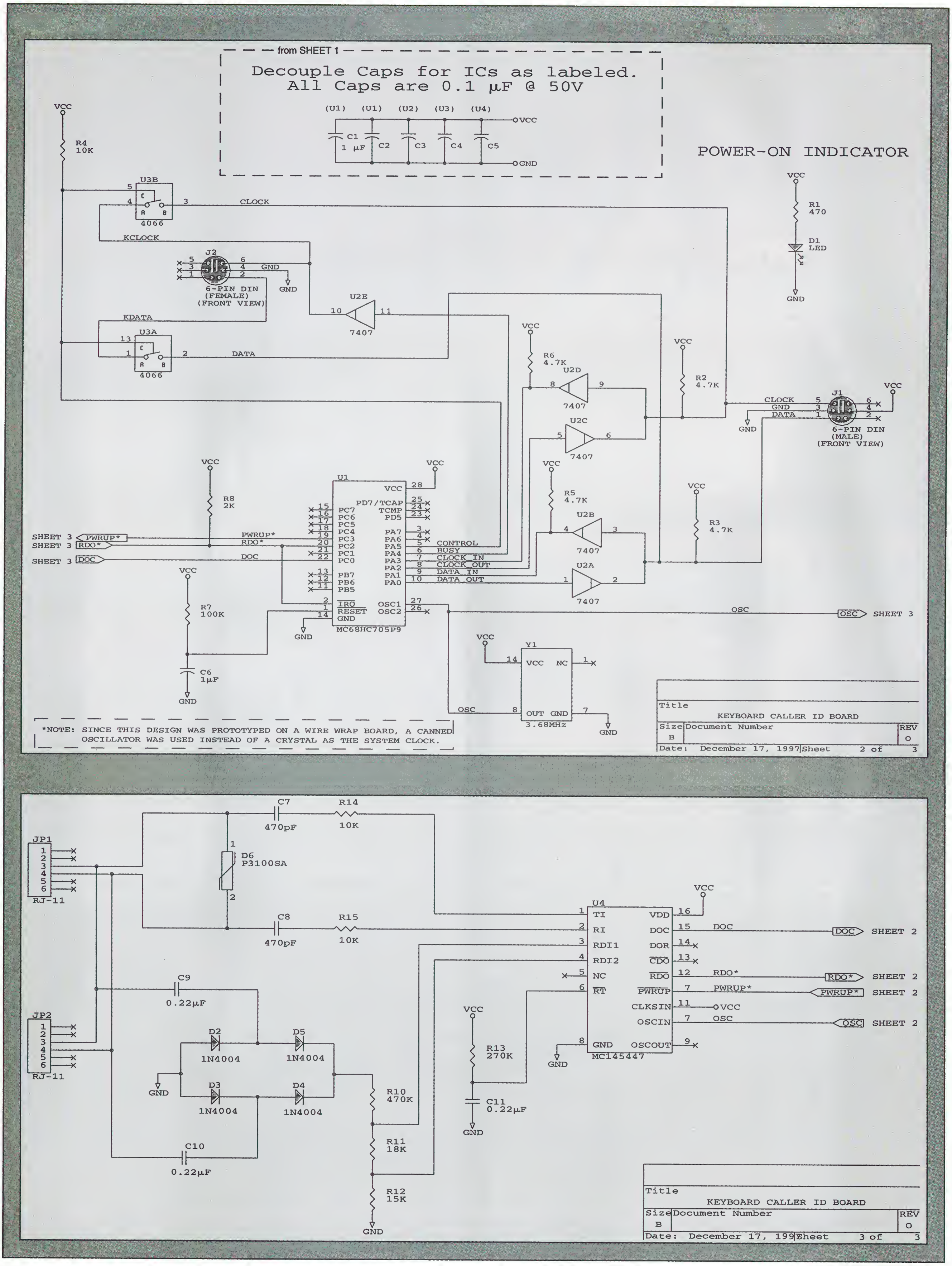


***Figure 1:** Keyboard Caller ID board schematics.*

device's power-up circuitry. To conserve power, the MC145447's power-up circuitry applies power to different sections of the device as they are needed.

In the power-up sequence, the /RT and RDI1 signals are used to activate power to the Ring Analysis section of the device. This section determines whether a valid ring signal is present on the twisted pair. As the schematics in Figure 1 shows, the voltage at the RDI1 pin is provided by resistor R10, which is part of a voltage divider circuit comprised of resistors R10, R11, and R12. (More complete schematics are available electronically in .EPS format; see "Resource Center," page 3.) The resistor network divides an AC coupled, rectified version of the voltage present between the tip and ring sides of the twisted pair into voltages that are sampled by the RDI1 and RD2 pins. The value of R10 is chosen such that if a voltage of 40Vrms or more is present on the twisted pair, which indicates that a power ring might be taking place, the RDI1 pin and its associated circuitry will turn power on to the Ring Analysis circuitry. The /RT is connected to a RC combination that holds the pin low during the low periods of a power ring. The RDI2 pin serves as the only input to the Ring Analysis section. The signal at this pin is provided by resistor R12 of the divider network. The duty cycle of this signal is used to validate the presence of a power ring. In the event that a power ring is detected, the Ring Analysis circuit asserts the /RDO pin.

The data-acquisition signals on the MC145447 consists of a Tip input (TI) and Ring input (RI) pin. The TI is AC coupled to the tip side of the telephone line's twisted pair through capacitor C7. The RI signal is AC coupled to the Ring side of the twisted pair through capacitor C8. The signal that is presented to these two pins is demodulated and converted into the digital stream that is output by the device.

In our application, the MC145447's interface with the system's microcontroller consists of three pins—the Data Out Cooked (DOC) pin, the /Ring Detect Out (/RDO) pin, and the /Power Up (/PWRUP) pin. The MC145447 outputs a digital stream on two pins, which are the Data Out Cooked (DOC) pin and the Data Out Raw (DOR) pin.

The DOR pin outputs the entire data stream demodulated by the device starting with the Channel Seizure and Mark Signals and ending with the checksum byte at the end of a transmission. The DOC pin, on the other hand, outputs data after a transmission passes an internal data validation process and does not output the Channel Seizure and Mark Signals. Data is captured by the MC68HC(7)05P9 by connecting DOC to pin PC3 on the MC68HC(7)05P9, which is configured as an input.

The /RDO pin is connected to pin PC2 of the MCU, which is configured as an input. As stated earlier, the /RDO pin is asserted when a valid power ring is detected on the twisted pair. The assertion of the /RDO pin, along with the start of the transmission of data within 0.5–1.5 seconds after the deassertion of /RDO, is used by the MC68HC(7)05P9 to qualify the start of a data stream from the MC145447.

The MC145447 has a requirement that its /PWRUP pin be at a Logic 1 for a minimum of 10µS after VDD reaches its full value. Typically, this requirement is met by delaying the assertion of /PWRUP with a RC circuit. To eliminate the need for these two components, the /PWRUP pin is connected to the MC68HC(7)05P9's PC3 pin, which is configured as an output. This pin asserts /PWRUP after an appropriate delay.

The Keyboard-Interface Block

The main function of the keyboard-interface block is to transmit Caller ID data captured from the MC145447 to an AT-compatible host computer through its keyboard interface. Pins PA0 and PA1 of the MC68HC(7)05P9 serve as the application's keyboard interface's data signal. PA0 is

configured as an output and is used to transmit data to the keyboard interface. PA1 works in conjunction with PA0 and is configured as an input. This arrangement satisfies the AT keyboard interface requirement that the keyboard interface data line be a bidirectional signal that is capable of both transmitting and receiving data to/from the host. Pins PA2 and PA3 function in a way that is similar to the PA0-PA1 pin pair. PA2 is configured as an output and generates the clock signal required for both keyboard-to-host and host-to-keyboard data transfers, and PA3 is confirmed as an input that reads the level on the clock line. Though the clock signal never functions as an input as does the data line, the AT keyboard interface protocol requires that its level be monitored in the event that the host wishes to transmit data to the keyboard. Since PA0 and PA2 are not open-collector outputs, they cannot be directly connected to the data and clock signals of the keyboard and keyboard interface. Therefore, a 7407 open-collector buffer serves as the interface between the MCU's keyboard interface signals and those of keyboard and the keyboard interface.

The design of the keyboard interface does not allow the keyboard to be connected to the interface while another device is transmitting to it. Therefore, the Caller ID device must disconnect the keyboard's clock and data signals from those of the keyboard interface whenever it transmits to the host. Port A pin PA5 is configured as an output and serves as the control signal for the 4066 analog switches that connect or disconnect the keyboard's signals to those of the interface. The number of tasks that a host computer's CPU may need to perform may prevent it from processing a scan code at the time that it is received at the keyboard interface. To prevent user keystrokes from being lost, the keyboard-interface protocol provides for a busy signal that the host sends to the keyboard to prevent it from sending scan codes until the host can process them. The host signals the keyboard that it is busy by holding the clock line low until it can accept new scan codes. While the host is busy, the keyboard stores the scan codes for new keystrokes in its internal buffer. To prevent the loss of any keystrokes that may be generated while the Caller ID device is transmitting to the host, the MC68HC(7)05P9 pulls the clock signal low after it disconnects the keyboard's signals from the interface. Port A pin PA5 is configured as a output and performs this function.

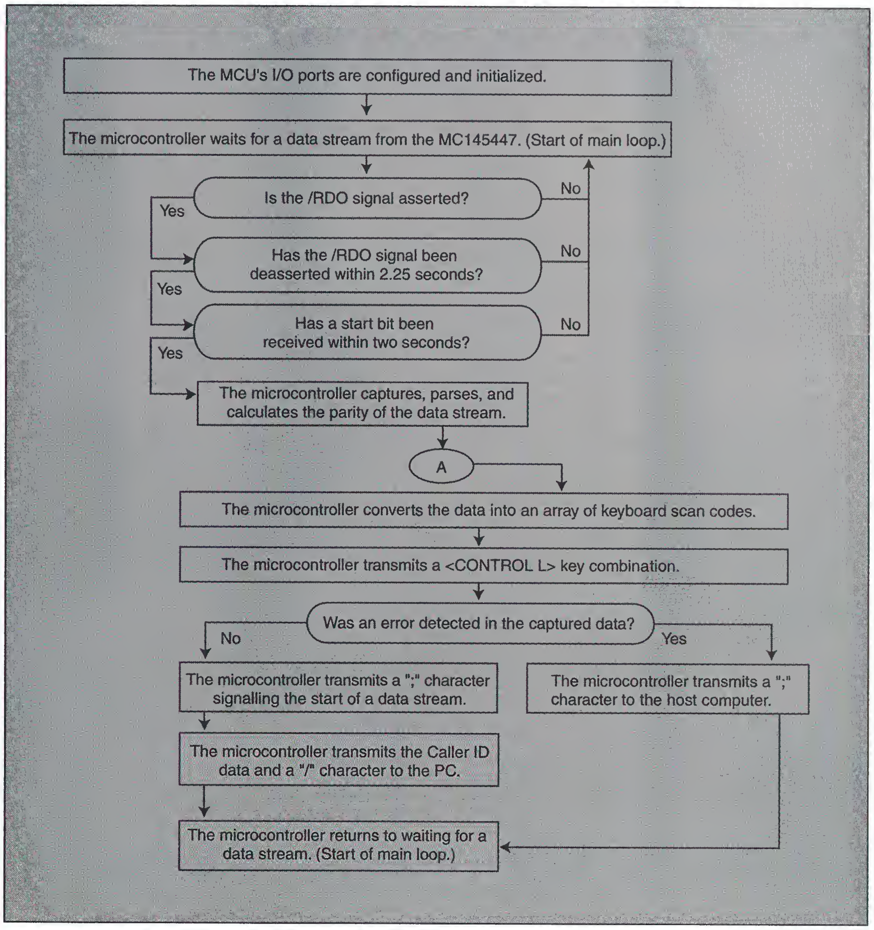


Figure 2: *Keyboard Caller ID device firmware flowchart.*

Keyboard Caller ID Device Software-Design Overview

The software design of this application is divided into two parts—the firmware that resides on the MC68HC(7)05P9 and CALLERID.EXE. The firmware's main function is to capture the raw digital data stream generated by the MC145447 and transmit it to the host computer for further processing (source code for the firmware is available electronically; see "Resource Center," page 3). Data is transmitted to the host in the form of keyboard scan codes that are sent through the host's keyboard interface. The host receives the scan codes and interprets them as keystrokes. The sequence of simulated keystrokes is read by CALLERID.EXE, which parses and converts the string back into binary data from which it extracts Caller ID information. CALLERID.EXE (source code for CALLERID.EXE is available electronically) then formats and displays the data in a pop-up dialog box. This division of functionality between the Caller ID device and the host computer allows for the greater portion of processing to be off loaded to the host computer where a larger amount of resources are available. This reduces the functionality of the Caller ID device thus allowing its design to be implemented with a smaller and cheaper microcontroller.

Keyboard Caller ID Device Firmware Design

As Figure 2 illustrates, the Caller ID device's firmware follows this program flow:

1. On reset, the general I/O pins on the MC68HC(7)P9 are configured and initialized to implement the Caller ID device's hardware design.
2. The firmware waits in a loop for the assertion of the MC145447's /RDO signal that is monitored on the MC68HC(7)05P9's PC2 I/O pin. The assertion of this signal indicates that a power ring has been detected on the twisted pair.
3. If the MC68HC(7)05P9 detects that the MC145447's /RDO pin is deasserted and a start bit on the DOC pin, the conditions are met for the MC68HC(7)05P9 to begin monitoring for a transmission.
4. The MC145447 transmits the CALLER ID data to the MC68HC(7)05P9 in the form of a raw digital stream on its DOC pin. The MCU reads the data from its PC0 pin.
5. On receiving the data from the MC145447, the MC68HC(7)05P9 parses the stream into individual bytes and checks the data for a parity error. If a parity error has been detected, it is flagged by a global variable, otherwise

(continued from page 76)
the data is converted into an array of AT keyboard scan codes for transmission to the host computer.
6. The application transmits a <CONTROL L> keystroke sequence as a series of scan codes. This interrupts the application that currently has the focus in Windows 95, activates CALLERID.EXE, and gives it the focus.
7. If a parity error was not detected during the reception of the CALLER ID data, the scan code array that represents the received data is transmitted to the host computer, otherwise an error code is sent.
8. The firmware returns to monitoring the twisted pair for a new Caller ID transmission.

The firmware's functions can be divided into three types of routines:

- Device initialization routines.
- Caller ID data-acquisition routines.
- Keyboard interface routines.

The device initialization routines configure and initialize the MC68HC(7)05P9's I/O pins to implement the application's hardware blocks. As mentioned earlier, Port A I/O pins PA0–PA5 are configured to implement the keyboard interface block, while three Port C pins, PC0, PCG, and PC3, serve as the MC68HC(7)05P9's interface to the MC145447. All remaining general-purpose I/O pins are configured as outputs to eliminate the need for pull-up resistors on them. The data acquisition routines of the firmware consists of the sampling and time delay routines that capture data from the MC145447's DOC line. The MC68HC(7)05P9 samples the data stream at its PC3 pin and parses it into individual bytes. The fact that each piece of Caller ID data begins with a start bit and ends with a stop bit, makes it easy to delineate between individual bytes. The time delay functions used for data acquisition routines are not only used to sample the bits within a byte but must also allow for the inter-character delays that the Interface allows.

The keyboard-interface firmware mainly consists of a transmission routine and its accompanying time delay functions. The keyboard interface's transmit function has within it a call to a routine that is capable of receiving host computer commands. If the host computer detects an error in the data that was sent to it by the keyboard, the host will hold the data low after bad transmission. The host will then send a Resend command (0xFE) to the keyboard requesting a retransmission of the data. Therefore, the Caller ID device must have a receive routine in the event that an error occurs.

For this application, the number of retransmission attempts was arbitrarily set at 1. Therefore, if an error occurs when the device sends a byte to the host, the device will capture the host's resend command and attempt a retransmission of the data. If the retransmission fails, the device will reconnect the keyboard's clock and data signals to those of the host and return to monitoring the telephone line. To transmit data to the host, the transmission routine toggles PA0, which is the data output signal, and the PA2 pin, which is the clock output signal, in accordance with the timing specifications for keyboard-to-computer data transfers. The host command reception routine reads the data from the PA1 pin and toggles the clock signal in accordance with the timing specifications for computer-to-keyboard data transfers.

CALLERID.EXE Design

CALLERID.EXE's design is divided into two parts — CALLERID.EXE (the executable program) and CALLDLL.DLL (the DLL containing the global hook function). Both modules were compiled with Microsoft Visual C++ Version 2.0. CALLDLL.DLL's code consists of a function to install the keyboard hook function and the hook function itself. In the code's call to the Windows API's *SetWindowsHookEx* function, the *idhook* parameter is set to WH_KEYBOARD, which is a predefined value that configures the hook function to handle keyboard events. This code is placed in a DLL because Windows 95 requires that global

hook functions reside in a DLL. The keyboard-hook function in this application must be global in scope so that CALLERID.EXE can be invoked regardless of what application may currently have the focus in Windows 95. The only limitation with CALLERID.EXE is that it will not be invoked if the current window with the focus is a DOS window.

The main function of the executable is to receive the Caller ID data from the Caller ID device, format it, and display it in a dialog box on the PC's monitor. As Figure 3 illustrates, the program flow of the executable is as follows:

1. CALLERID.EXE is invoked immediately after Windows 95 boots up. The main window of the CALLERID application is initialized to come up in the hidden state. This causes CALLERID.EXE to begin executing in the background of Windows 95.
2. CALLERID.EXE accesses CALLERID.DLL and installs the keyboard hook function into the Windows 95 stream. The hook function now examines each keystroke that is entered by the user for the <CONTROL L> hotkey sequence.
3. On detecting a <CONTROL L> key combination, the keyboard hook function calls the Windows API *FindWindow()*, function to locate the application's hidden main window. The Windows *ShowWindow()* function is then called to activate CALLERID.EXE's main window and give it the focus in Windows 95.
4. CALLERID.EXE displays a pop-up dialog box on the monitor displaying the text: "Receiving Data…".
5. The application waits for a keystroke from the Caller ID device.
6. If CALLERID.EXE receives a ";" character from the Caller ID device, the device has detected a parity error in the Caller ID data received from the telephone line. The CALLERID.EXE will then display "Line Error" in the dialog box: Otherwise, it acquires the full stream of Caller ID data from the device.
7. C-string manipulation functions are used to parse the string into the two character segments that represent each byte of Caller ID data. C string conversion functions are then used to convert each ASCII segment into the original binary data that was captured on the Caller ID device.
8. CALLERID.EXE formats the binary data so that it can be displayed in the dialog box. CALLERID.EXE will format data according to whether the Caller ID data received is in the SDMF or MDMF format.
9. The Caller ID information is displayed in the dialog box. The dialog box remains displayed until users press one of the box's OK or Deactivate buttons.
10. The dialog box is hidden again if the user presses the OK button. CALLERID.EXE then returns to waiting for a hot key sequence. If the Deactivate button is pressed, CALLER.EXE will be deactivated and will no longer function until Windows 95 is reset.

Keyboard Caller ID Device Operating Instructions

To use the Keyboard Caller ID system we've presented here:

1. Copy CALLERID.EXE to the hard drive and directory of your choice. A suggested path might be: C:\CALLERID\.
2. Copy CALLDLL.DLL to the C:\WINDOWS\SYSTEM\ directory.
3. Add CALLERID.EXE to the Windows 95 Start Menu.
4. Disconnect the keyboard's connector from the host computer's keyboard port.
5. Connect the Keyboard Caller ID device to the host computer's keyboard interface.
6. Connect the keyboard's connector to the receptacle for it on the Keyboard Caller ID device.
7. Connect the telephone line to one of the R-J11 connectors on the Keyboard Caller ID device.
8. Connect a telephone extension line between the Keyboard Caller ID's second

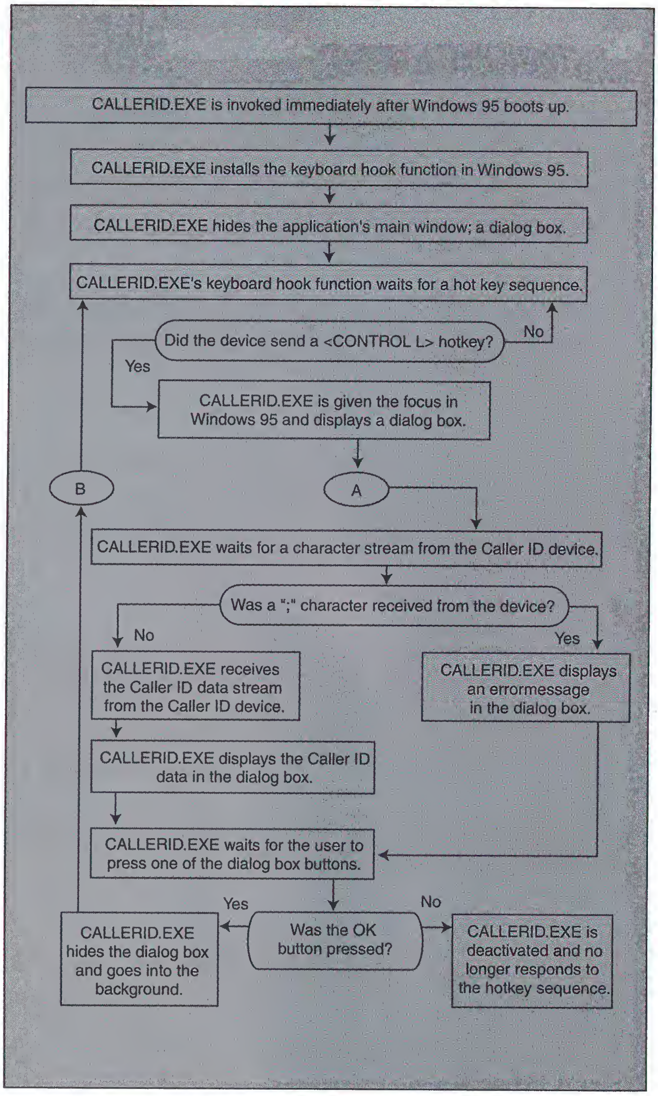


Figure 3: *CALLERID.EXE program flowchart.*

R-J11 connector and your telephone. This completes the hardware installation of the Keyboard Caller ID device.

9. Shut down and restart Windows 95.
10. Caller ID should now activate. Caller ID will display a dialog box with Caller ID information every time a valid transmission is received. To deactivate the program, press the Deactivate button in the dialog box.

References

Holzner, Steve. *Advanced Visual C++ 4*. M&T Books, 1996.

Konzak, Gary J. *PC Keyboard Design*. Second Edition, Annabooks, 1993.

Motorola MC68HC705P9 Technical Data Specification.

Motorola MC145447 Technical Data Specification.

Messmer, Hans-Peter. *The Indispensable PC Hardware Book: Your Questions Answered*. Addison-Wesley, 1994.

LSSGR: Voiceband Data Transmission Interface Section 6.6, GR-30-CORE, Issue 1, Bellcore. December 1994.

"Message Type Values for SDMF and MDMF." *Digest of Technical Information* Bellcore. May 1996.

DDJ

Rendering XML Documents Using XSL

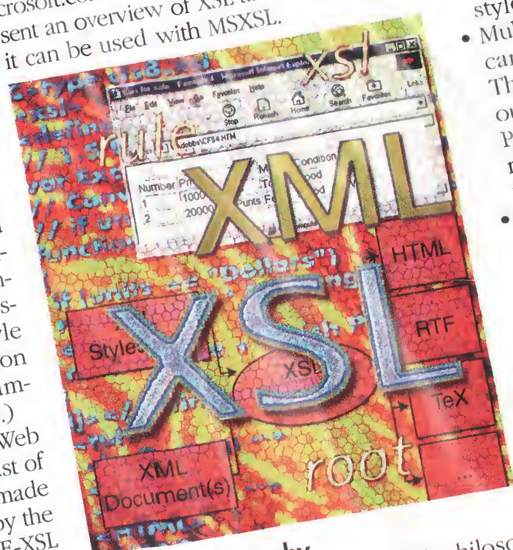
...eping content ...d format separate

Sean McGrath

Central to the eXtensible Markup Language (XML) philosophy is that the structure and content of information should be captured without concern for how the information will be rendered on a computer display, paper, voice synthesis, and others. Responsibility for rendering XML has been delegated to a sister standard known as eXtensible Style Language (XSL). (For more information on XML, see my article "XML Programming in Python," *DDJ*, February 1998.)

Like XML, XSL is a World Wide Web Consortium (W3C) initiative. In August of 1997, a draft proposal for XSL was made available as a discussion document by the W3C (http://www.w3.org/TR/NOTE-XSL.html). Although, the working draft for XSL is just that, a number of XSL applications have already appeared. In particular, Microsoft has released MSXSL, a "technology preview" implementation that is freely available at http://www.microsoft.com/xml/. In this article, I will present an overview of XSL and illustrate how it can be used with MSXSL.

Sean, chief technical officer and cofounder of Digitome Electronic Publishing (http://www.digitome.com/) is a member of the World Wide Web Consortium's XML Special Interest Group. He is also the author of ParseMe.1st: SGML for Software Developers *(Prentice-Hall, 1997) and* XML By Example: Building E-commerce Applications *(Prentice-Hall, 1998). Sean can be reached at sean@digitome.com.*



The XSL Philosophy

As Figure 1 illustrates, the XSL philosophy can be summed up as "late binding of presentation semantics." In simple English, the idea is that information about how a document should look when rendered (presentation semantics) is separated from the document content and housed in a stylesheet. The process of creating a rendition of the content happens late — preferably right at the point that someone wants to view it (hence, late binding).

This late binding approach has some significant benefits:

- The look and feel of a document (or thousands of documents) can easily be changed simply by changing the stylesheet.
- Multiple renditions of the same content can be created from a single source. These renditions can include different output notations such as RTF, HTML, or Postscript. They can involve rearrangements of the content, creating multiple views of the information.
- The information content is "future proofed." Creating a new rendition to a new notation (or a notation yet to be invented), is a matter of applying the necessary stylesheet.
- Keeping the content free of rendering information makes it easier to process the content. That is, searching, harvesting, or rearranging the content can be performed without worrying about how the formatting information is intermingled with the content.

There are a number of core concepts that are central to XSL, including:

Flow Objects. In XSL, the process of transforming an XML document into a notation such as RTF, HTML, or Postscript, is expressed in terms of the construction of flow objects, which are

Sample #3: Accessing Attribute Values

The *Car* elements in Figure 2 have attributes for price and currency information. These can be accessed in XSL by escaping to the ECMAScript scripting language. XSL provides an *eval* element that can be used to house script code. In Listing Three(a), the rule is modified to let *Car* elements access the attribute information. Listing Three(b) is the HTML from this modified stylesheet.

The CDATA section in the *eval* element is an XML construct that shields text from the attentions of the XML parser. The CDATA section begins with the "<![CDATA[" string and ends at the "]]" string. It is a good idea to use CDATA sections to shield script code, since characters such as "<" and "&" can have special meanings to an XML parser.

Sample #4: Creating a Table

Listing Four(a) is a stylesheet creating a simple HTML table layout of "car for sale" information. Listing Four(b) is the result of applying this stylesheet to the XML file.

- All attribute values for the HTML flow objects must be quoted to make the stylesheet well-formed XML. The correct way to specify a table with a border is <TABLE BORDER="1">. The syntax <TABLE BORDER=1>, which is HTML valid, generates a parsing error in MSXSL; see Example 1.

- The first cell in each table row contains the relative number of the *Car*—1, 2, and so on. This number is automatically generated by the *childNumber()* function. The built-in XSL functions for automatic numbering are useful in creating stylesheets.

Figure 3 shows what the generated HTML file looks like in Internet Explorer 4.0.

Sample #5: Rearranging Content

With XSL, it is possible to exert control over the order in which elements in the source document are processed. This allows document content to be selected and rearranged prior to creating the output. In Listing Five(a), a table of car maker names is created; Listing Five(b) shows the result of applying this stylesheet to the XML file.

Only the *Maker* element data has appeared in the output. This is because the *select-element* element indicates that only the *Maker* children of *Car* elements are processed. By default, the *select-element* element looks at the children of the current element to find matches. It is also possible to arrange for *select-element* to

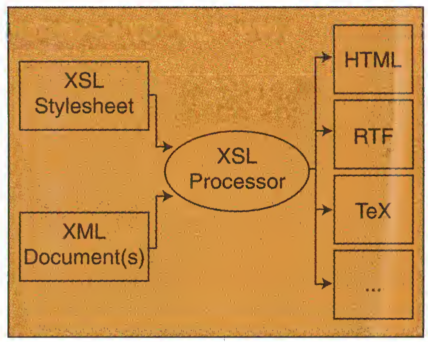


Figure 1: *The XSL philosophy.*

```
C>type cars.xml
<?xml version = "1.0"?>
<CarsForSale>
    <Car Price = "10000" Units = "Dollars">
    <Maker>Toyota</Maker>
    <Condition Type = "Good"/>
    <Color>Red</Color>
    </Car>
    <Car Price = "20000" Units = "Irish Punts">
    <Maker>Ford</Maker>
    <Condition Type = "Good"/>
    <Color>White</Color>
    </Car>
</CarsForSale>
```

Figure 2: *Typical XML document.*

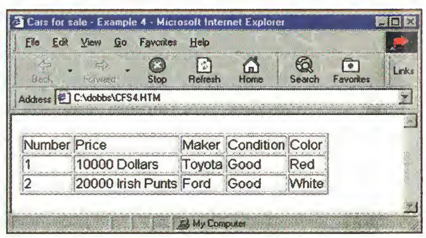


Figure 3: *Typical generated HTML file viewed using Internet Explorer 4.0.*

pages, columns, paragraphs, table cells, and so on.

Platform-Independent Flow Objects. XSL specifies a set of standard flow objects such as paragraph, page sequence, table, and the like. Using these platform-independent flow objects lets you create multiple output notations with a single XSL stylesheet. The type of notations that can be created is limited only by the back-end notations supported by the XSL processor. Strong candidates for XSL back ends include RTF, FrameMaker MIF, and TeX.

HTML-Specific Flow Objects. To facilitate the use of XSL stylesheets to generate HTML, XSL provides a set of HTML-specific flow objects. Given the vast amount of HTML-aware software in existence, it makes sense to use this software, while simultaneously retaining the advantages of XML over HTML as a data representation.

Construction Rules. Flow-object construction in XSL is controlled by rules in the XSL stylesheet. These rules specify what flow objects are to be created and what they should contain. Flow objects can be thought of as containers for document content and/or other flow objects creating a tree-like hierarchy known as a "flow-object tree." Flow-object construction rules take the form of a pattern and action. The pattern part specifies the conditions under which the rule triggers. The action part specifies what flow objects to construct.

Characteristics. Flow objects can have associated characteristics that differ depending on the type of flow object being constructed. A paragraph flow object, for example, might have margin and tab characteristics. A table cell might have border and spanning characteristics. The characteristics to be applied to flow objects can be controlled in the XSL stylesheet by means of style rules. Style rules take the same general form as construction rules, and consist of pattern and action components.

Scripting. No stylesheet language that provides a fixed set of rendering capabilities can provide all the processing power needed. There comes a point where a "Turing Complete" programming language is the best way to get the job done. The XSL draft specifies ECMAScript (a standardized version of JavaScript—ECMA 262) as a built-in scripting language. A number of mechanisms are provided in XSL for escaping to ECMAScript to perform calculations, define functions, and so on.

Introducing MSXSL

MSXSL is Microsoft's technology preview implementation of the XSL draft specification. Don't confuse it with MSXML, which is Microsoft's implementation of an XML parser. Indeed, MSXSL uses MSXML to parse and load XSL stylesheets.

MSXSL focuses on creating HTML from XML and, for the time being, only supports HTML flow objects. The simplest way to use MSXSL is via the provided command-line utility that takes the input XML file (-i), input stylesheet file (-s), and output HTML file (-o). For example, the command *C>msxsl -i foo.xml -s foo.xsl -o foo.htm* processes the foo.xml file with respect to the foo.xsl stylesheet specification, then generates the foo.htm output file.

MSXSL is Microsoft's technology preview implementation of the XSL draft specification

To illustrate how to use XSL and MSXSL, I'll return to the XML document (see Figure 2) presented in my February 1998 article.

Sample #1: Getting Started

Listing One(a) (listings begin on page 97) creates a simple stylesheet to convert the XML document in Figure 2 to HTML. Some things to note about this stylesheet:

- It is an XML document and uses a set of element types—*xsl, rule, root,* and so on—defined by the XSL language. The tags for these elements appear in lowercase.
- It signifies the creation of HTML flow objects by using HTML tags—BODY, TITLE, and so on. These tags appear in uppercase.
- It consists of a single flow-object construction rule. The pattern that triggers the rule is the root element (*<root/>*).
- The children element (*<children/>*) tells the XSL processor that all the children of the element that triggered the construction rule should be processed, and the results of processing these elements should be inserted into the output flow-object tree.

Listing One(b) is the result of processing the XML document with this stylesheet. While it's hardly the world's most exciting HTML file, there are some important things to note:

- All the data content of the XML document (the content of the *Maker* and *Color* elements) has found its way into the output document. The default in XSL is that the content of elements that do not trigger construction rules simply flows over to the output document at the point where it is encountered.
- The attribute values (*Price*, *Units*, and *Type*) do not appear in the output document. This is also the result of the default behavior of XSL.

Sample #2: Rudimentary Formatting

Listing Two(a) adds a few more construction rules to create slightly more pleasing HTML output, while Listing Two(b) presents the result of processing the XML document with this stylesheet. Things to note about this stylesheet and the resultant HTML include:

- The pattern part of the flow-object construction rules use the *target-element* element, which can be used in a variety of ways to specify context-sensitive rules. Here, I used the simplest form in which the *target-element* is an empty element (denoted by the slash in *<target-element/>*).
- The same construction rule can be triggered for multiple element types by specifying multiple empty *target-element* elements. I've used this to cause the same rule to trigger on *Condition* and *Color* elements. The paragraph generated for the *Condition* element is empty because attribute values are not, by default, included in the output document.
- The stylesheet is an XML document and, thus, must be well-formed XML. This is why the HR flow object uses XML syntax to indicate it is an empty element (*<HR/>*).

```
C>msxsl -i cfs.xml -s cfs4.xsl -o cfs4.htm
Error in style sheet 'cfs4.xsl'
ParseException: Expecting name instead of '1'
Location: file:///C:/DOBBS/cfs4.xsl (9,17)
Context: <xsl><rule><HTML><BODY><TABLE>
```

***Example 1:** MSXSL parsing error.*

search all descendants by specifying the value *"Descendants"* to the optional *from* attribute: *<select-elements from = "Descendants">*. The *<children/>* element used in previous examples is shorthand for *<select-elements from = "Children">*

Sample #6: Scripting

As a final example, the stylesheet in Listing Six(a) uses ECMAScript to present all prices in Irish Punts in the generated HTML; Listing Six(b). The *define-script* element is used to create global variables and functions, and the *eval* element is used to invoke functions and access global variables.

XSL and CSS

Although Cascading Style Sheets (CSS) can be used to render XML documents, XSL provides many more capabilities than CSS. With CSS, the document structure is essentially fixed and is simply mapped onto the available flow objects. With XSL, the document structure can be rearranged and can be processed multiple times. For example, with XSL is it possible to perform a traversal to generate a table of contents, then perform a second traversal to render the content proper. Also, XSL is programmable via ECMAScript, thus providing a Turing Complete environment in which to create rendering effects.

(On the other hand, CSS is simple and familiar to many HTML users, and work is underway at Hewlett-Packard to create an extended implementation of CSS, known as "Spice," which makes up for some of these deficiencies.)

XSL and DSSSL

XSL draws heavily on the concepts used in the Document Style and Semantics Specification Language ISO 10179 (DSSSL) Standard for SGML rendering. The DSSSL Standard uses a programming language based on Scheme as its expression language. Many of the DSSSL designers have been instrumental in the design of XSL, and work is underway to make the XML-based expression language of XSL formally a part of the DSSSL international Standard. Henry Thompson of the University of Edinburgh has developed the XSLJ conversion utility that converts XSL specifications into DSSSL specifications. These can then be used with implementations of the DSSSL Standard such as Jade (http://www.jclark.com/).

Conclusion

XSL is an important part of the overall vision of XML. The core XML effort has three separate strands—XML itself, XML rendering (XSL), and XML hypertext linking (XLL). The speed with which companies such as Microsoft have moved to implement XSL has come as something of a surprise to many. Even in its current basic state, MSXSL is capable of real work and provides a glimpse of the capabilities you can expect in the next generation of web browsers, which should process and render XML directly.

DDJ

(Listings begin on page 97.)

For More Information

XSL Draft Specification: http://www.w3.org/TR/NOTE-XSL.html

Microsoft's MSXSL XSL Processor
Technology Preview: http://www.microsoft.com/xml/

XSL Discussion List and Archive: http://www.mulberrytech.com/xsl/xsl-list/

XSL-J (XSL to DSSSL Conversion Utility): http://www.ltg.ed.ac.uk/~ht/xslj.html

Jade DSSSL Engine: http://www.jclark.com/

DSSSL Discussion List and Archive: http://www.mulberrytech.com/dsssl/dssslist/

Examining the Dragon Speech-Recognition System

Speech enabling your Windows apps

Al Williams

Remember the movie *2001: A Space Odyssey*? One of the most memorable characters in that movie was, of course, "Hal" the computer. Like most movie computers (the exception being the Macintosh in *Star Trek IV*), Hal was smart and spoke with humans. Well, it isn't quite 2001, but it's getting pretty darned close, and still, real computers don't speak and understand as well as their movie counterparts. However, recent developments have brought a plethora of voice-recognition tools for Windows. Though these tools are mainly for dictating into word processors, as a developer you can certainly think of other applications. Wouldn't it be nice to piggyback your program on an existing voice recognition system? You can, using toolkits such as Dragon Systems' DragonXTools.

DragonXTools visual controls make it possible for you to speech-enable Windows applications with up to 60,000-word dictation using Visual Basic, Visual C++, and other development tools that support VBX/OCX controls or C-callable libraries. Users then use DragonDictate for Windows to enter text, data, and commands into Windows apps simply by speaking. You can distribute the DragonXTools custom controls royalty free. However, you'll need run-time licenses to distribute DragonDictate for Windows.

In this article, I'll show how to use DragonXTools custom controls to add speech recognition to programs. In doing so, I'll use Visual Basic 5 to write a voice-activated autodialer (available electronically; see "Resource Center," page 3). Since the controls are ActiveX controls, however, you can use most any language with them.

Al's most recent book is MFC Black Book *(Coriolis, 1997). He can be contacted at http://www.al-williams.com/.*

How Does it Work?

Even if you do not take special action, DragonDictate still works with your program, although it may not work well, depending on how your program is written. When DragonDictate sees your program running, it scans your menu items and control captions. Since DragonDictate generates its own speech models, it can respond to users speaking your menu items and control captions. Of course, some captions work better than others. Sometimes DragonDictate can't determine an appropriate speech model. Also, if you use custom controls for graphical buttons, DragonDictate can't decide what that means.

DragonDictate operates in either command or dictate mode. Usually, DragonDictate is in command mode, which lets you speak menu commands, or key names. In command mode, you can operate the computer without using a mouse or a keyboard. However, entering text in command mode is a chore. You'd have to speak each letter individually. Instead, when you want to enter text you say "dictate mode." Now, DragonDictate interprets user speech as words until you say the command "command mode." (One problem is that users must remember to enter dictation mode. A speech-aware program might handle this automatically.)

In short, it isn't strictly necessary to alter your programs in any way to make them work with DragonDictate. However, with DragonXTools, you can get significant benefits. For example, you can set your own pronunciations for words that DragonDictate might misinterpret (or graphics that DragonDictate can't read at all). You can also respond to words with your own actions, use DragonDictate's macro language, and share the sound system with DragonDictate (many sound cards can only listen or talk at one time).

The DragonXTool Toolkit

DragonXTools includes a control that lets you recognize speech and interact with the Dragon engine, and another that converts plain text into speech. This works about as you expect; the speech sounds computer-generated, which is not always pleasing.

Using the components isn't difficult and the manual provides examples to help you get started. The examples are for VB, but the manual includes advice on how to use the components in C++, Delphi, and Java as well.

Dragon separates words it will recognize into vocabularies and groups. It scans different vocabularies depending on the current situation. The Dragon speech control lets you manage vocabularies and groups. You can create new words, and control which groups Dragon examines for speech recognition.

Like all ActiveX controls, the Dragon control has properties, methods, and events. Table 1 is a list of the members used in the VB program I present here. Many functions that you'll need to use require you to access Dragon's scripting language (via the *Script* property).

Using the controls is straightforward once you get the hang of it. You first have to make sure DragonDictate is running. If it isn't, you can start it before your program proceeds using something like Example 1. Once DragonDictate is running, you have to attach your speech control to the speech recognition engine. You can do this with the *Attach* property. Then

(continued from page 86)
you are ready to begin creating your own vocabularies. You can set the currently active vocabulary and group using the *Vocabulary* and *Group* properties. When Dragon recognizes any of the words in your control's group, it sends an event so that you can act on the word.

Name	Type	Description
DDWinShutDown	Event	DragonDictate Shut Down.
MicChanged	Event	Microphone turned on/off.
UserChanged	Event	User changed.
SpeechRecognized	Event	Dragon recognized a word in the control's current group.
SpeechActionComplete	Event	Dragon completed all actions for a recognized word (in any active group).
SpeechRejected	Event	Dragon couldn't recognize a word or phrase.
Version	Property	Version number.
Attach	Property	Sets to True to attach to the Dragon engine.
EventsEnabled	Property	Lets you control the delivery of events to your application.
Group	Property	Set current group.
Vocabulary	Property	Sets current vocabulary.
GroupCaption	Property	Sets text in voice bar.
GroupFirst	Property	Gets first group as a prelude to iterating through the groups in a vocabulary.
GroupNext	Property	Continues iterating through the groups (see GroupFirst).
WordFirst	Property	Gets first word in a group as a prelude to iterating through the words in a group.
WordNext	Property	Continues iterating through words.
IgnoreInDictation	Property	Set to True if you want the Dragon engine to ignore your active words while in dictate mode.
Microphone	Property	Lets you turn the microphone on, or disable it (which frees it and the associated sound system for other tasks).
Repetitions	Property	Sets default number of training iterations used for training.
Script	Property	Setting this property causes the Dragon engine to execute commands in its script language.
TrainGroup	Property	Lets you train an entire group; although DragonDictate doesn't strictly require training, it can improve accuracy.
WordSelection	Property	Sets which words in a group get training in response to a TrainGroup command.
TrainWord	Property	Lets you train a single word (see TrainGroup).
Voicebarheight	Property	Sets the height of the voice bar (the pop up window that DragonDictate uses to interact with users).
VoicebarLeft	Property	Left side of the voice bar.
VoicebarTop	Property	Location of the top of the voice bar.
VoicebarWidth	Property	Width of voice bar.
AddGroup	Method	Adds a new group.
DeleteGroup	Method	Deletes a group.
FindGroup	Method	Determines if a group exists.
AddVocabulary	Method	Adds a new vocabulary.
DeleteVocabulary	Method	Deletes a vocabulary.
Find Vocabulary	Method	Determines if a vocabulary exists.
AddWord	Method	Adds a new word and associates a macro with it.
DeleteWord	Method	Deletes a word.
FindWord	Method	Determines if a word exists.
Listen	Method	Temporarily adds a group to the list of active groups.
GetLastErrorCode	Method	Returns last error code.
IsDDWinRunning	Method	Returns True if DragonDictate is running.
StartDDWin	Method	Start DragonDictate.
CloseDDWin	Method	Shuts down DragonDictate.

Table 1: DDSpeech *control members.*

Problems

Although the tools are generally easy to use, I did find a few things I wish had been different. First, DragonDictate doesn't work well under Windows NT. The Dragon web site (http://www.dragonsystems.com/) has a FAQ about this. You can use Dragon and the tools under NT, but the behavior is quirky. Occasionally, you'll be thrown into another window, for example. Worse, during development, the system frequently crashes and hangs VB. This seemed to have something to do with breakpoints, so I don't expect it would be a problem in the shipped program, but it sure made writing software a chore.

Another thing I thought odd is the way Dragon handles sleeping. If you are like me, you can't really leave your microphone on all the time for DragonDictate to listen. There are phones ringing, dogs barking, and all manner of other noises in my office. Suppose you are dictating text into a program and the phone rings. You can say "go to sleep," which puts Dragon in a dormant state. It still listens, but it doesn't do anything until you say "wake up." However, DragonDictate still notifies you when it recognizes any of your words. That means you have to know if Dragon is asleep or not. However, you can't ask Dragon if it is asleep. If you want to handle this problem, you have to define your own "go to sleep" and "wake up" commands and do the work yourself. Then you'll still get the events for other words, but your program will know it should be asleep. The Dragon manual has examples of several ways to do this.

The Design

I've always thought the voice-activated autodialers on some high-end telephones are a great idea. You just speak a name into the phone and it dials away. (And before you ask, yes, you can get a headset that works with Dragon and your phone.) For the first cut, I tried to make the program understand the name I was saying and I figured I would store the phone numbers in a flat file or database.

As I got the speech part working, I realized I really didn't need a database, since I could store the phone number and name along with the speech model. Of course, you don't want to have to say the name and the number to dial on the phone, right? That defeats the whole purpose. However, Dragon lets you specify alternate pronunciations for words, as I'll show you in a bit. Since users have their own vocabulary, that means individual users also have a private phone book.

Figure 1 shows a completed dialer application. You don't need any special code or controls to handle the three buttons—Dragon takes care of them automatically. But you do need some special work to make the phone automatically dial when you speak a name.

When you say "add" or click the Add button, the program brings up a simple form that lets you make a new entry. The form has two fields, *Name* and *Number*, that you can jump to by simply saying the

```
If Not IsDDWinRunning() Then ' Dragon not running
  If Not StartDDWin() Then ' Didn't start up either
     MsgBox "Can't start DragonDictate"
  End If
End If
```

Example 1: *Starting up* DragonDictate.

appropriate word. Also, when the *Name* field has the focus, the program automatically places Dragon in dictation mode. It is impressive how many common names Dragon correctly interprets.

Saying "number" (or clicking the button with that name) brings up an input box. From here you can just say numbers aloud to dial them. Say "okay" when you are done, or "cancel" to abort. Dialing the phone is easy with an MSCOMM ActiveX control. The dialer assumes you have a Hayes-compatible modem on COM1 (although that's easy to change).

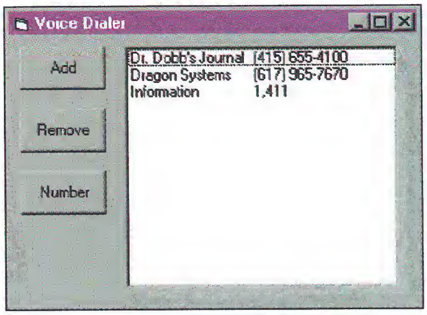


Figure 1: *The voice dialer application.*

Implementation

The trick to this program is setting up word recognition. When you add a new name, the program constructs a string to "teach" Dragon. The string consists of the name, a tab character, and the phone number. However, this would be awkward to pronounce, so the program also adds a square bracket, the name alone, and closing bracket. By placing alternate text in brackets, you are telling Dragon that the text between the brackets is the correct pronunciation for the preceding word. The program then feeds this string to the *AddWord* method of the speech control. If Dragon can't deduce a speech model for the word, *AddWord* returns False and the program lets the user train the word in question. Listing One (listing begins on page 98) includes this logic (see the *Add_Click* subroutine).

Interestingly, the listbox holds names in the same format (but no pronunciation in square brackets). This makes it easy to create a single *Dial* routine that handles a string from the voice recognition or listbox. It also makes it easy to reconstruct the listbox from the vocabulary data on startup (see the *Form_Load* subroutine in Listing One).

The *DDSpeech1_SpeechRecognized* routine handles the voice dialing. The only reason there is more than one line of code in this routine is that I wanted to change the listbox selection to reflect the dialed number. Visual feedback is important when you are dealing with voice command, because voice is not 100 percent accurate. When you delete a name, for example, the program is careful to prompt you before taking action. It might be a good idea to add a similar safeguard to the dialing routine, too.

The form used to add names is available online. There, you can find the code that sets Dragon's mode when each text box receives the focus. This allows users to dictate names without having to explicitly set the dictation mode.

Other Possibilities

Once you have the ability to work with voice commands and dictation, there are many other ways you can make your application more voice friendly. For example, by using the *SetHomeGroup* script command, you could restrict the phone-number fields to accept only words that make sense for phone numbers. You can also use *DgnTTS* control to convert words back to voice (although for simple uses, you might be better off just playing prerecorded wave files).

Although DragonXTools has some problems (poor NT compatibility and difficult to manage sleep mode), it is exciting to watch a program respond to spoken words. Probably the biggest disadvantage is that users have to already have one of the Dragon products that provide the actual speech processing. Of course, if you are building a dedicated system, or you are willing to license the product from Dragon, this may not be a problem. Just try to resist the urge to speak into your mouse.

For More Information

Dragon Systems Inc.
320 Nevada Street
Newton, MA 02160
617-965-5200
http://www.dragonsystems.com/

DDJ

(Listing begins on page 98.)

JDBC Drivers and Web Security

Security and success go hand-in-hand

Mukul Sood

The Java Database Connectivity (JDBC) specification is an interface for database programming. At present, there are at least two dozen commercially available JDBC implementations. As I described in "Examining JCBC Drivers" (*DDJ*, January 1998), JDBC driver vendors have considerable freedom in their implementations of the JDBC specification. Still, to be JDBC compliant, JDBC drivers must support the ANSI SQL 2 Entry Level standard, pass JavaSoft conformance tests, and fall into one of four categories as defined by JavaSoft:

- Type I, JDBC/ODBC Bridge.
- Type II, Native-API, Partly Java Driver.
- Type III, Net-Protocol, All Java Driver.
- Type IV, Native-Protocol, All Java Driver.

I won't detail the differences between these four categories, referring you instead to my January 1998 article. In this article, I'll focus on the security features offered by various driver vendors, and describe how they fit in with the various application deployment architectures.

Java, Security, and the Web

Although security is not officially part of the JDBC specification, it is nonetheless critical for many database applications deployed on the Web. The success of online trading, shopping marts, and sensitive retail transfer information (credit cards, bank accounts, social-security numbers, and the like) over an inherently insecure medium (the Internet) depends on reliable security. Consequently, JDBC driver vendors are beginning to offer security features such as encryption and authentication, which can be used by applications to deliver secure and reliable services.

Mukul is a system architect at Digital Focus. He can be contacted at mukuls@digitalfocus.com.

Applications can be deployed on the web in various configurations.

- Client behind a firewall. Firewall security is an integral part of security infrastructure at many organizations. This especially holds true if the organization allows Internet access. Typically, organizations put a firewall between the private trusted network and the public Internet. In this case, access would be allowed only to certain protocols (HTTP) and at a specific port. Thus a client behind a firewall would be able to connect through the firewall, if it uses these allowed protocols. Figure 1 shows this configuration. If the JDBC client speaks a certain protocol, it would be able to connect to the web-server host only if it supports tunneling of that protocol through HTTP packets, assuming the firewall on the client allows HTTP traffic to get through; Sybase's jConnect, for example, uses Tabular Data Stream format (TDS). (For more on Java and secure tunneling, see "Java Q&A," by Kenneth Golomb and Thomas Sorgie, *DDJ*, June 1998.) The other possibility would be for the client-side firewall to allow the protocol (TDS, SqlNet, and so on) to get through.
- Client outside of a firewall. Another commonly used configuration; the client accesses the Internet without going through a firewall. In this case, the client can use vendor-specific protocols to connect to the server host. The server could be running on the same host as the web server that served the applet or it could be on a different host. The server could be a gateway server (in case of Type IV drivers) or a JDBC server (in case of Type III drivers). The JDBC server could be set up on a host different from the web server, in which case it would be necessary to run a proxy server on the web-server host. This proxy would pass through JDBC client requests to the JDBC server and response from the server to the client; see Figure 2.
- Client in an intranet. A private trusted network would have no firewalls installed. Examples would be department- or company-wide intranets that are sealed off from public networks (such as Internet). In this configuration, the client would connect to the web-server host, which also runs the gateway server for Type IV drivers, the proxy server for Type III drivers, or runs the JDBC server for Type III drivers.

  The role of a gateway for a Type IV driver is different from that for a Type III driver. A gateway for a Type IV driver is actually a connection server that manages database connections. In case of a Type III driver, the gateway is basically a proxy for the actual connection server. A Type IV driver can only connect to the web server from which it was downloaded so it can open a database connection to the web-server host. Consequently, the database server would need to be installed on the same host as the web server. The gateway gets around this limitation by taking the role of a connection server.

These are some common deployment configurations. Many configurations are possible, given existing infrastrucure, application requirements, IT standards employed by the organization, and other factors.

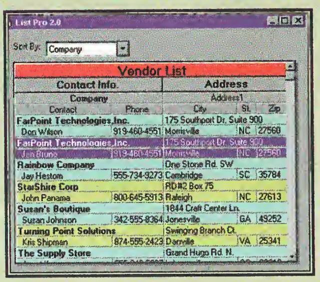
The Secure Sockets Layer Protocol

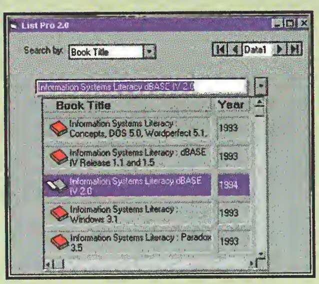
Encryption and authentication are integral parts of security, and the Secure Sockets Layer (SSL) protocol supports both of these features. SSL is the dominant protocol for encrypting general communications between browsers and servers. It is built into both the Netscape and Microsoft browsers. The SSL protocol operates at the TCP/IP

FarPoint's LIST PRO

List Pro contains the most powerful list box and combo box. These extremely robust controls allow you to display up to two billion list items, display single records or items on multiple rows, merge cell text for easier viewing, search for specific list items, or sort list items using unlimited keys. Together, these features make List Pro the Ultimate Data Viewer.

The Ultimate Data Viewer





GENERAL FEATURES

Include up to 2 billion list items in your control ● Select single or multiple list items ● Display data in single or multiple columns ● Use ReadOnly mode to create a powerful database viewer that allows you to scroll quickly through data using the arrow keys or the mouse ● Develop for the Internet or Intranet with digitally-signed ActiveX controls ● Use the Property Designer to easily access the features of the list box or combo box at design-time (VBX and ActiveX only)

CUSTOM APPEARANCE

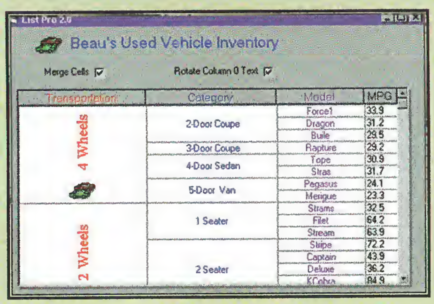
Insert bitmaps or icons into any column or row ● Assign foreground and background colors to any cell, row, column, header, or group ● Customize column headers ● Customize the application of 3-D effects for almost all parts of the control ● Display a drop shadow on the control at all times or only when the control receives focus ● Display flat or 3-D lines between columns, rows, or both ● Change the orientation of text (horizontal, vertical LTR, vertical RTL, rotate down, rotate up, or invert)

LIST BOX/COMBO BOX

Select, deselect, or delete rows ● Insert or delete columns and groups ● Allow the user to drag and drop single columns or groups of columns ● Hide entire columns or lock columns from being resized at run time ● Choose from three combo box styles ● Assign different pictures to the selected and unselected states

HEADER GROUPING

Create multiple levels of column or group headers ● Define any number of columns or groups as children of other groups ● Allow a single record or a group of data to span across multiple rows ● Merge column cells that contain identical text for easier data viewing



DATA BINDING

Use the Virtual Data Manager in bound or unbound mode to display large data files quickly ● Use Visual Basic to bind separate data controls to the combo box's drop-down list and edit portion ● Control which data fields are bound to individual columns

FarPoint technologies

Order today by calling 800-645-5913 x3025 or download a trial version at WWW.FPOINT.COM

FARPOINT TECHNOLOGIES INC.

Main 919-460-4551 *Sales* 800-645-5913 x3025 fpsales@fpoint.com

(continued from page 90)
transport layer, one level below application-specific protocols such as NNTP (news), HTTP (web), and SMTP (mail). Any program that uses TCP can be modified to use secure SSL connections by making a few source code changes. Figure 3 shows the SSL protocol stack.

The main trade-off for putting SSL at the transport layer is that it is not specifically tuned for HTTP and therefore may not always be efficient for web browsing. A minor limitation is that an SSL connection must use a dedicated TCP/IP socket. SSL uses cryptographic principles such as digital envelope, signed certificates, and message digests. An important feature of SSL is flexibility with regard to choices of encryption algorithm (symmetric algorithms like DES, 3DES, RC2, RC4), message digests (MD5, SHA), and authentication methods (RSA public keys and certificates or Diffie-Hellman key exchange).

The combination of encryption algorithm, message-digest function, and authentication is known as a "cipher suite." The Netscape browser supports more than 30 cipher suites. Likewise, JDBC drivers support various cipher suites.

When an SSL client first makes contact with a server, the two negotiate a common cipher suite. In general, the two try to pick the strongest encryption methods that they have in common.

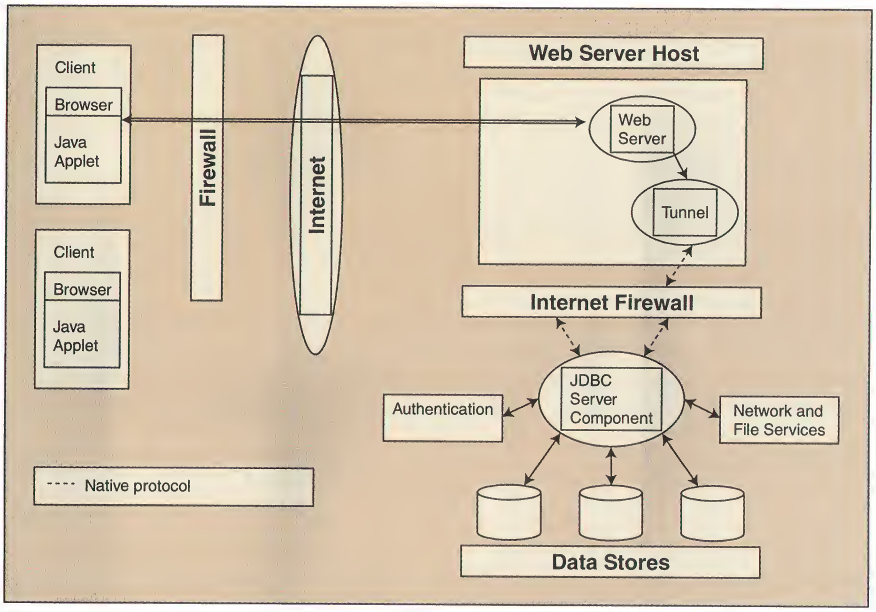


Figure 1: *Java client behind a firewall accessing database through a Type III JDBC driver.*

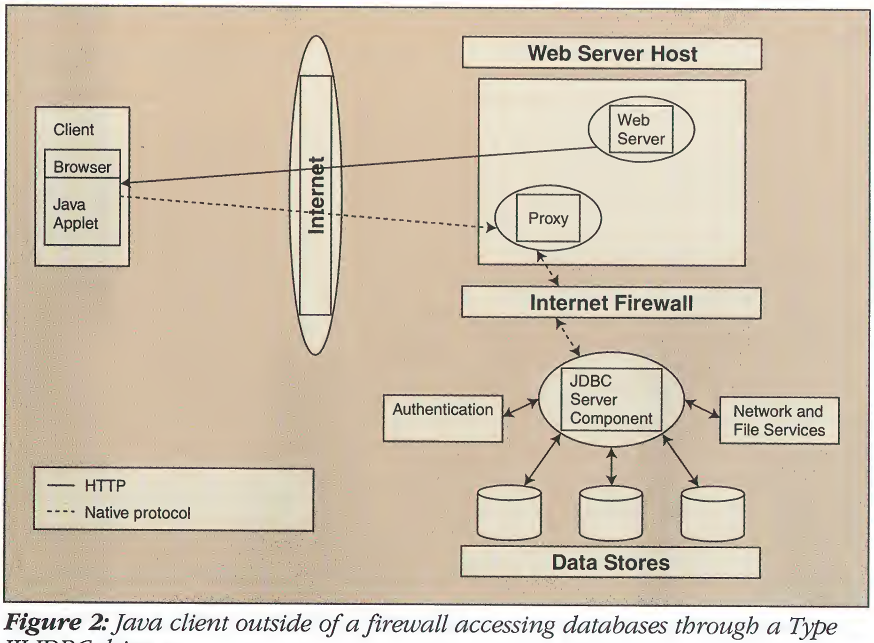


Figure 2: *Java client outside of a firewall accessing databases through a Type III JDBC driver.*

When an SSL connection is in place, all browser-to-server and server-to-browser communications are encrypted including the URL of the requested document, contents of the requested document, and contents of the HTTP header.

Table 1 summarizes the drivers and shows what features are offered by each vendor.

Tengah/JDBC

Weblogic's Tengah/JDBC is a Type III implementation of JDBC for use with Java applets and applications. It includes features such as authentication and encryption (through Tengah SSL), access to name services (NIS, NDS, and the like), access-control lists, HTTP, IIOP tunneling, and proxy support.

Tengah/JDBC secures networked applications with optional encryption, authentication, and authorization based on RSA SSL, X.509 certificates, and access-control lists. It provides encryption and authentication through Tengah SSL, which is Weblogic's implementation of SSL Version 3. Tengah SSL uses RSA as the key exchange method. It supports both server-to-client and two-way authentication. User authentication is provided through X.509 certificates. Tengah supports message protocols such as t3 (Tengah's native protocol, which uses TCP sockets) and HTTP. The SSL layer permits these protocols to be secured as t3s and HTTPS. When a t3s connection is opened, a certificate exchange occurs that guarantees the server's identity. Thereafter, communications are encrypted. In the case of an applet, the certificate used to establish the HTTPS connection is used for the applet's t3s connection.

Tengah uses the Java Naming and Directory Interface (JNDI) to provide access to underlying directory-naming services that support LDAP, such as Novell's NDS and Sun's NIS.

It uses the JDK 1.1 ACL Principal, Group, Permission, Acl, and AclEntry interfaces in its implementation of ACL. Tengah's support for ACLs follows the Javasoft javasecurity.acl specification. The ACL support built into Tengah, the Weblogic Realm, depends upon users having an entry in a list that grants or denies permission to access a particular service. Tengah organizes Users, Groups, Permissions, and Acls into Realms which map to Principal, Group, Permission, Acl, and AclEntry. A Tengah application can set up its own ACL, and its own Realm. The Weblogic Realm, which is supplied as a Tengah service, provides ACLs for Tengah.

Tengah also supports HTTP and IIOP tunneling; thus, a client behind a firewall could connect to the Tengah server through these tunneling mechanisms.

SequeLink

SequeLink, Intersolv's Type III driver, provides features such as multiple levels of security, encryption, and a lightweight Java proxy.

SequeLink supports six levels of security: DBMS system authentication, host system authentication, data encryption, application authentication, ReadOnly, and Autocommit.

SequeLink interfaces with the host system security provided by the OS system security (Windows NT or UNIX, for instance), delivering an additional layer of user authentication for access to the system services. SequeLink does not support JNDI in the current version. SequeLink uses encryption on-the-wire for all data and user account information (including UID/PWD). The current version does not use SSL. Application authentication allows system administrators to determine specific client applications that can access specific SequeLink services. Client applications pass a specific key that the SequeLink server then validates as an authenticated server configuration. Certain apps (query tools) can thus be restricted to read-only while others can have full write and update permissions.

SequeLink offers the unique ReadOnly and Autocommit filters. When Autocommit is switched on, the server begins and commits each statement immediately, regardless of application setting. When ReadOnly is switched, the server processes only SQL SELECT statements and does not send statements such as INSERT, UPDATE, and DELETE to the database engine. This feature can be useful for organizations where custom applications are used for order entry (requiring end users to have write permissions), but the same end users may need to run query tools for decision support or data mining. If a user were to accidentally change a value in these other tools, then the entire data could be corrupted.

Another SequeLink feature is its Java proxy, which is essentially a Java application running on the web-server host, its role being that of a pass-through server to the SequeLink JDBC server.

Product	Vendor	Version	Driver Type	Encryption	Authentication	SSL Support	ACL	Proxy Support	HTTP Tunneling
Tengah	Weblogic	3.0	3	Yes	Yes	Yes	Yes	Yes	Yes
SequeLink	Intersolv	3.5	3	Yes	Yes	No	Yes	Yes	No
JSQL/Ingres	Caribou Lake Sw		3	Yes	Yes	No	Yes	Yes	Yes
SqlRetriever	SCO		3	No	Yes	No	Yes	Yes	No
jConnect	Sybase	3.0	4	Yes	Yes	Yes	NA	Yes	Yes
Oracle Driver	Oracle	1.0	4	No	No	No	NA	Yes	No
Fast Forward	Connect Software	3.0	4	Yes	Yes	No	NA	Yes	Yes
OPENjdbc	I-Kinetics		3	Yes	Yes	Yes	Yes	Yes	0

***Table 1:** Security features offered by various JDBC drivers*

SequeLink does not have HTTP tunneling support in its current version. Clients behind a firewall that allows only HTTP traffic, therefore, would not be able to connect to the server.

JSQL/Ingres

JSQL (Java to SQL) is a Type III driver from Caribou Lake Software for connecting to Ingres databases. It has various components that provide proxy services, authentication, tunneling, and the main server component that connects to database servers. These components are:

- Jsvr, a JDBC connection server.
- Jpass, a proxy server for Jsvr.
- Jtunnel, for HTTP tunneling.
- Jauth, validates username/password on remote computers in network.

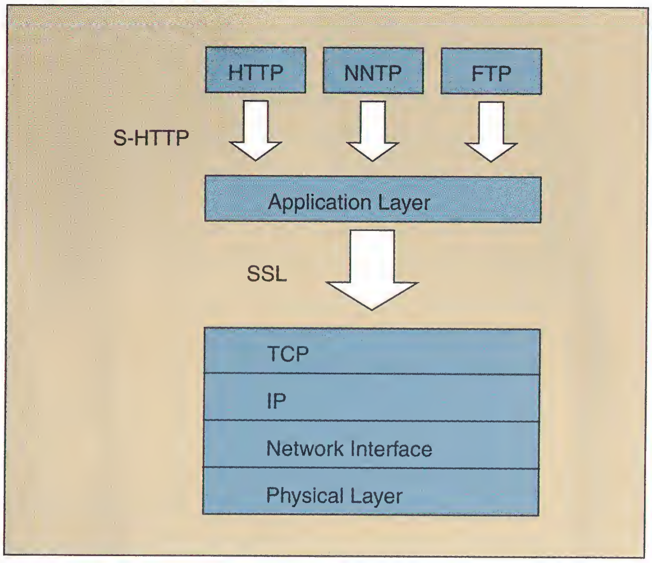


***Figure 3:** The SSL protocol stack.*

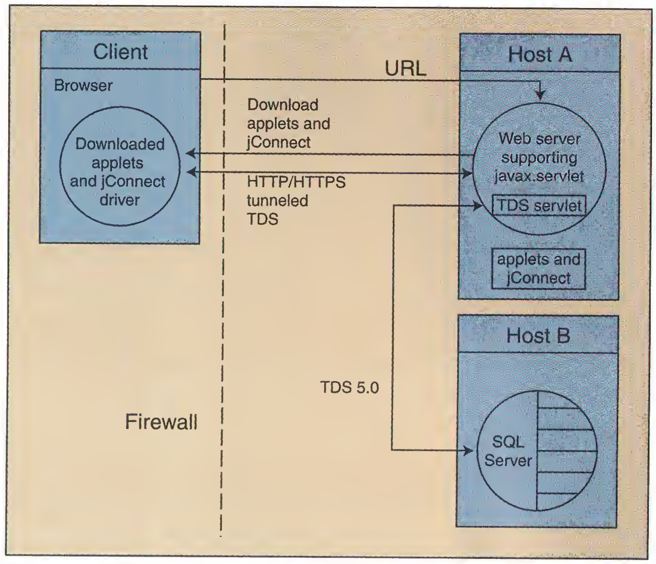


***Figure 4:** jConnect configuration for a client behind a firewall.*

Jtunnel is an HTTP tunneling server that tunnels JDBC connections through firewalls and reverse proxy servers. Jtunnel could also be used with a secure HTTP server to provide encryption between the Java client and the web server. Jtunnel comes with a CGI script named "Jcgi" that should be installed on the web-server host if tunneling is required. This script takes the HTTP packet and sends it to Jtunnel, which retrieves the data from the packet and forwards it either to Jpass or to Jsvr depending on the connection URL in the request. Jpass provides pass-through services; Jsvr can be installed on a different host than the web server, and Jpass would relay client requests to Jsvr.

Jauth is the authentication server for authenticating JDBC client connects to the Jsrv connection server. It supports authentication methods such as operating-system authentication (for server OSes), database authentication (Open Ingres only), and remote authentication (server OSes). Jsvr can be configured with an access-control list.

SQL Retriever

SQL Retriever is SCO's server module, with which the SCO JDBC client communicates. The JDBC client allows Java applets or applications to access information from Oracle, Informix, Ingres, Sybase, Interbase, and Progress databases on any UNIX platform.

The driver communicates with the server module, which is installed either on the web server or another host. The driver communicates with the server using Sun RPC (Remote Procedure Call) mechanism. The RPC layer forms a connection using Java sockets. The SQL-Retriever server is a UNIX daemon process. The default configuration for JDBC client and server is that the client gets the port of the server from portmapper and the server is started as root and registers itself with the portmapper. If the server is behind a firewall, the port 111 (used by portmapper) is blocked. In this case, the server can be made to listen on a specific port (start the server with a –l flag, flag should be followed by the port number). At the client side, the port should be specified in the JDBC connect URL, the client then will not use the portmapper, it will go straight to this port. If the client is behind a firewall which allows only HTTP traffic, then the client would not be able to connect to the server.

SQL Retriever's security features include a security manager which lets system administrators control user access to the database or to specific tables in the database, and a proxy RPC-Reflector, which allows servers to be installed on a host other than a web server (it passes through client requests to the server). SQL Retriever does not offer encryption and server-authentication.

jConnect

Sybase's jConnect is a Type IV driver for connecting to the Sybase SQL Server. For connecting to servers using SSL, jConnect Release 3.0 provides a Java servlet that should be installed on the web server that hosts the applet. The web server should support the javax.servlet interfaces, which enables jConnect to support encryption using the web server as the gateway. jConnect uses the TDS-tunnelled HTTP protocol in proxy and firewall configurations; see Figure 4.

With the TDS-tunneling servlet, requests from a client to the back-end server that go through the gateway include a GET or POST command, the TDS session ID (after the initial request), back-end address, and status of the request as query parameters for the request. The TDS stream is contained in the body of the request. Two header fields indicate the length of the TDS stream included in the request packet, and the session ID assigned by the servlet. When the client sends in an initial request (login request), the servlet creates a session ID, strips the HTTP headers and sends the TDS data to the server, gets the results back from the server in TDS format, assembles an HTTP packet (passing the session ID in the header), and sends the packet to the client. Subsequent requests from the client contain that session ID in the header.

Oracle ThinJDBC

ThinJDBC, Oracle's Type IV driver, has a footprint of about 150 KB and provides its own implementation of a TCP/IP version of Oracle's SQLNet/Net8 protocol. This driver only works with TCP/IP-based networks. It supports Oracle databases Versions 7.2 and upward. ThinJDBC does not have proxy support; it can, however, be used with the Oracle Connection manager to achieve three-tier configurations. Since it does not support HTTP tunneling, it will not work with those firewalls that only allow HTTP through. It would work with firewalls that allow SQLNet traffic. The communication between an applet that uses the ThinJDBC driver and the database happens over Java TCP/IP sockets. The connection can only be made if the web browser (where the applet is executing) allows a socket connection to be made. In JDK 1.0.2-based browsers (such as Netscape 3.0), the applet would only be able to open a connection to the host from which it was downloaded. In this case, the database would have to be present on the same host as the web server.

...th JDK 1.1-based browsers, this re-...iction does not apply if the applet is ...gned.

...Oracle Connection Manager may be de-...loyed in combination with JDBC applets ...o provide secure access to Oracle envi-...ronments. Connection Manager incorpo-...rates a Net8 application proxy, which lets ...system administrators control how a con-...nection request gets routed. Through the ...use of rules, requests may be filtered based on parameters such as:

- Destination or Origin IP address.
- Oracle System Identifier (SID).
- Data encryption/security preferences.

The ThinJDBC applet can connect to a Connection Manager running on the web-server host and have the Connection Manager redirect the packets to an Oracle server running on a separate host.

Fast Forward

FastForward, a Type IV driver from Connect Software, provides Java clients with direct access to Microsoft SQL Server (all versions) and Sybase SQL Servers (Versions 4, 9, 10, and 11). FastForward works by directly transferring and receiving information from Java to SQL Server using TCP/IP sockets. The format of data passed back and forth is TDS. Version 3.0 offers features such as xencryption and HTTP tunneling through the FastForward Security Proxy. FastForward Proxy is a pure Java application that provides connectivity between clients on the Internet and servers within your network. It also provides HTTP tunneling, compression, and encryption. Proxy uses symmetric private key encryption in 16 rounds with 64-bit key for encryption. ...also supports DES.

OPENjdbc

OPENjdbc is I-Kinetics CORBA-based Type III driver. The driver communicates to the DataBroker server through IIOP protocol. The databroker server is based on CORBA, using Iona's Orbix Object Request Broker to provide features such as multithreading, connection pooling, and load balancing. OPENjdbc invokes methods and services defined in the DataBroker's IDL files. Figure 5 illustrates the OPENjdbc architecture.

OPENjdbc driver offers security features such as SSL encryption and authentication (through Orbix SSL) and HTTP tunneling of IIOP. The SSL option is provided through Orbix SSL, which allows Orbix- and Orbix-Web-based applications to be easily retrofitted with SSL security. Orbix SSL replaces the default IIOP protocol with the standardized SSL-IIOP protocol, which is essentially IIOP over secure SSL connections. The SSL option provides authentication using public-key cryptography (RSA, DSS) and encryption using block-encryption methods (DES, RC4).

The DataBroker server need not be on the same host as the web server that served the applet. An IIOP proxy on the web server can route all requests to the databroker. Iona's Wonderwall offers this feature along with its firewall features such as examination, filtering and logging of IIOP requests, HTTP tunneling support, and ACL support.

Conclusion

Security is an important aspect of applications that deal with sensitive data and as the ... firewall). JDBC driver ve... which could meet the security needs ... these applications. Your choice of software should be made based on factors such as application requirements, security needs, deployment configuration, expandibility, and standards supported by the vendor.

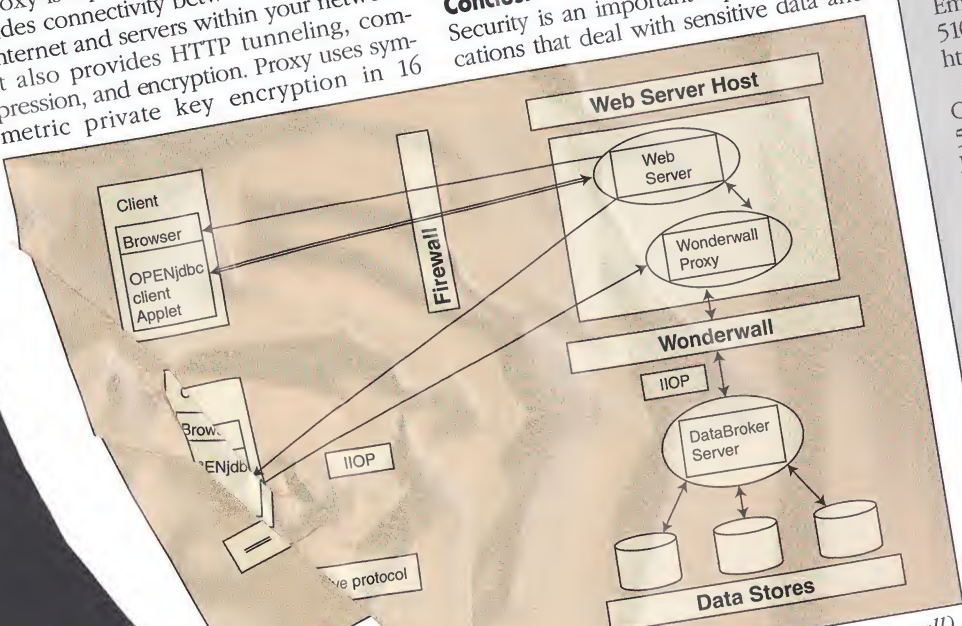


Figure 5: The ... accessing Dat... ...lient behind firewall, client outside a firewall) ...erver is behind Wonderwall.

For More Information

WebLogic Inc.
417 Montgomery Street
San Francisco, CA 94104
415-659-2600
http://www.weblogic.com/

Intersolv Inc.
9420 Key West Avenue
Rockville, MD 20850
301-838-5000
http://www.intersolv.com/

Caribou Lake Software Inc.
4780 Beacon Hill Road
St. Paul, MN 55122
612-688-9470
http://www.cariboulake.com/

SCO
425 Encinal Street
P.O. Box 1900
Santa Cruz, CA 95061-1900
408-425-7222
http://www.sco.com/

Sybase Inc.
6475 Christie Avenue
Emeryville, CA 94608
510-922-3555
http://www.sybase.com/

Oracle Corp.
500 Oracle Parkway
Redwood Shores, CA 94065
650-506-7000
http://www.oracle.com/

Connect Software Inc.
81 Lansing Street
San Francisco, CA 94105
415-543-6695
http://www.connectsw.com/

I-Kinetics Inc.
17 New England Executive Park
Burlington, MA 01803
781-270-1300
http://www.i-kinetics.com/

DDJ

```
typedef struct
{
    float x, y, z;  /* coordinates */
} make_vertex_list;
```

Listing Two

```
typedef struct
{
    int a, b, c;        /* array indices */
    int a_s, a_t,       /* (s, t) texture coordinates */
    b_s, b_t,
    c_s, c_t;
} make_index_list;
```

Listing Three

(a)

```
(vertex_list[index_list[Ø].a].x, vertex_list[index_list[Ø].a].y,
                                 vertex_list[index_list[Ø].a].z)
```

(b)

```
(vertex_list[index_list[Ø].b].x, vertex_list[index_list[Ø].b].y,
                                 vertex_list[index_list[Ø].b].z)
```

(c)

```
(vertex_list[index_list[Ø].c].x, vertex_list[index_list[Ø].c].y,
                                 vertex_list[index_list[Ø].c].z)
```

Listing Four

```
typedef struct
{
    make_vertex_list *vertex;
} make_frame_list;
```

Listing Five:

(a)

```
frame_list[F].vertex[index_list[P].a].x
frame_list[F].vertex[index_list[P].a].y
frame_list[F].vertex[index_list[P].a].z
```

(b)

```
frame_list[F].vertex[index_list[P].b].x
frame_list[F].vertex[index_list[P].b].y
frame_list[F].vertex[index_list[P].b].z
```

(c)

```
frame_list[F].vertex[index_list[P].c].x
frame_list[F].vertex[index_list[P].c].y
frame_list[F].vertex[index_list[P].c].z
```

Listing Six

(a)

```
R: m_palette_buffer [ m_pixel_buffer[Ø]]
G: m_palette_buffer [ m_pixel_buffer[1]]
B: m_palette_buffer [ m_pixel_buffer[2]]
```

(b)

```
R: m_palette_buffer [3 * m_pixel_buffer[P]+Ø]
G: m_palette_buffer [3 * m_pixel_buffer[P]+1]
B: m_palette_buffer [3 * m_pixel_buffer[P]+2]
```

(c)

```
R: m_palette_buffer [3 * m_pixel_buffer[X + Y*Width]+Ø]
G: m_palette_buffer [3 * m_pixel_buffer[X + Y*Width]+1]
B: m_palette_buffer [3 * m_pixel_buffer[X + Y*Width]+2]
```

Listing Seven

```
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

isting Eight

```
lEnable(GL_CULL_FACE);
lEnable(GL_TEXTURE_2D);
PolygonMode (GL_FRONT, GL_FILL);
TexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
```

ting Nine

```
byte *unScaled = new GLubyte [m_iWidth * m_iHeight * 4];
 (j = Ø; j < m_iHeight; j++) {
r (i = Ø; i < m_iWidth; i++) {
unScaled[4*(j * m_iWidth + i)+Ø] =
      (GLubyte) m_palette_buffer[3*m_pixel_buffer[j*m_iWidth+i]+Ø];
nScaled[4*(j * m_iWidth + i)+1] =
      (GLubyte) m_palette_buffer[3*m_pixel_buffer[j*m_iWidth+i]+1];
Scaled[4*(j * m_iWidth + i)+2] =
      (GLubyte) m_palette_buffer[3*m_pixel_buffer[j*m_iWidth+i]+2];
Scaled[4*(j * m_iWidth + i)+3] = (GLubyte) 255;
```

Ten

```
ate memory for the new rescaled texture */
 = new GLubyte [m_iscaledWidth * m_iscaledHeight * 4];
}
/* use the OpenGL function to rescale */
gluScaleImage (GL_RGBA, m_iWidth, m_iHeight, GL_UNSIGNED_BYTE, unScaled,
     m_iscaledWidth, m_iscaledHeight, GL_UNSIGNED_BYTE, glTexture);
/* reclaim memory of the unscaled texture */
delete [] unScaled;
```

TEXTURE MAPPING

Listing One

```
void Draw_Triangle(float xØ,float yØ,float x1,float y1,
                             float x2,float y2, int color)
{
// this function rasterizes a triangle with a flat bottom
// compute left side interpolant
float dx_left = (x2 - xØ)/(y2 - yØ);
// compute right side interpolant
float dx_right = (x1 - xØ)/(y2 - yØ);
// seed left and right hand interpolators
float x_left = xØ;
float x_right = xØ;
// enter into rasterization loop
for (int y=yØ; y<=y1; y++)
    {
    // draw the scanline
    Draw_Line(x_left, x_right, y, color);
    // advance interpolants
    x_left+=dx_left;
    x_right+=dx_right;
    } // end for y
} // end Draw_Triangle
```

Listing Two

```
// initialize u,v interpolants to left and right side values
ui = ul;
vi = vl;

// now interpolate from left to right, i.e, in a positive x direction
for (x = xstart; x <= xend; x++)
    {
    // get texture pixel value
    pixel = texture_map[ui][vi];
    // plot pixel at x,y
    Plot_Pixel(x,y,pixel);
    // advance u,v interpolants
    ui+=du;
    vi+=dv;
    } // end for x
```

DVD

Listing One

```
// grab an interface to the Annex J methods
hr = m_pgraph->QueryInterface(IID_IDvdControl, (void **) &m_pUserOperations);
if ( !(FAILED(hr)) )
{
    // start playing title 1, chapter 3
    hr = m_pUserOperations->ChapterPlay( 1, 3 );
    // view the 2nd angle
    m_pUserOperations->AngleChange( 2);
    // turn off annoying foreign language subtitles
    m_pUserOperations->SubpictureStreamChange( 1, FALSE );
    // release interface
    m_pUserOperations->Release();
}
```

Listing Two

```
switch (Event)
    {
      case EC_DVD_BUTTON_CHANGE        :
        // lParam1 contains number of active buttons
        // if there are ANY buttons alive--enable butto... ation
        if ( lParam1 > Ø )
        {
            bActiveButtons = TRUE;
        }
        break;
    case EC_DVD_ANGLE_CHANGE           :
              // lParam1 contains the current viewin...
        break;
    case EC_DVD_TITLE_CHANGE           :
       // lParam1 contains the currently playing titl...
       break;
    case EC_DVD_CHAPTER_START          :
       // lParam1 contains the currently playing c...
       break;
   case EC_DVD_VALID_UOPS_CHANGE :
     // lParam1 contains the currently activ...
     break;
```

Listing One

(a)

```
C>type cfs1.xsl
<!-- Ultra simple XSL stylesheet -->
<xsl>
 <rule>
 <!-- Pattern -->
 <root/>
 <!-- Action -->
 <HTML>
 <HEAD>
 <TITLE>Cars for sale - Example 1</TITLE>
 </HEAD>
 <BODY>
 <children/>
 </BODY>
 </HTML>
 </rule>
</xsl>
```

(b)

```
C>msxsl -i cfs.xml -s cfs1.xsl -o cfs1.htm
C>type cfs1.htm
<HTML>
<HEAD>
<TITLE>Cars for sale - Example 1</TITLE>
</HEAD>
<BODY>
ToyotaRedFordWhite
</BODY>
</HTML>
```

Listing Two

(a)

```
<!-- Process Car elements by processing all children and then adding
a horizontal rule -->
<rule>
 <!-- Pattern -->
 <target-element type = "Car"/>
 <!-- Action -->
 <children/>
 <HR/>
</rule>
<!-- Process Maker elements by prefixing some literal text and then
processing all children -->
<rule>
 <!-- Pattern -->
 <target-element type = "Maker"/>
 <!-- Action -->
 <P>
 Make of Car: <children/>
 </P>
</rule>
<!-- Process both Condition and Color elements in the same way--simply
create HTML paragraphs -->
<rule>
 <!-- Pattern -->
 <target-element type = "Condition"/>
 <target-element type = "Color"/>
 <!-- Action -->
 <P>
 <children/>
 </P>
</rule>
```

(b)

```
C>msxsl -i cfs.xml -s cfs2.xsl -o cfs2.htm
C>type cfs2.htm
<HTML>
<HEAD>
<TITLE>Cars for sale - Example 2</TITLE>
</HEAD>
<BODY>
<P> Make of Car: Toyota
</P><P>
</P><P>
Red
</P><HR><P> Make of Car: Ford
</P><P>
</P><P>
White
</P><HR>
</BODY>
</HTML>
```

Listing Three

(a)

```
<rule>
 <target-element type = "Car"/>
 <P>
 Price = <eval><![CDATA[
 getAttribute("Price") + " " + getAttribute("Units")
 ]]></eval>
 </P>
 <children/>
 <HR/>
</rule>
```

(b)

```
C>msxsl -i cfs.xml -s cfs3.xsl -o cfs3.htm
C>type cfs3.htm
<HTML>
<HEAD>
<TITLE>Cars for sale - Example 3</TITLE>
</HEAD>
<BODY>
<P> Price = 10000 Dollars
</P><P> Make of Car: Toyota
</P><P>
</P><P>
Red
</P><HR><P> Price = 20000 Irish Punts
</P><P> Make of Car: Ford
</P><P>
</P><P>
White
</P><HR>
</BODY>
</HTML>
```

Listing Four

(a)

```
C>type cfs4.xsl
<xsl>
 <rule>
    <!-- Pattern -->
 <root/>
 <!-- Action -->
 <HTML>
 <HEAD>
 <TITLE>Cars for sale - Example 4</TITLE>
 </HEAD>
 <BODY>
 <TABLE BORDER="1">
 <TR>
 <TD>Number</TD>
 <TD>Price</TD>
 <TD>Maker</TD>
 <TD>Condition</TD>
 <TD>Color</TD>
 </TR>
 <children/>
 </TABLE>
 </BODY>
 </HTML>
 </rule>
<rule>
<!-- Pattern -->
 <target-element type = "Car"/>
 <!-- Action -->
 <TR>
 <!-- Automatically number the table rows -->
 <TD><eval>childNumber(this)</eval></TD>
 <TD>
```

(continued on page 98)

(continued from page 97)

```
<eval>getAttribute("Price") + " " + getAttribute("Units")</eval>
</TD>
<children/>
</TR>
</rule>
<rule>
<!-- Pattern -->
<target-element type = "Maker"/>
<target-element type = "Color"/>
<!-- Action -->
<TD>
<children/>
</TD>
</rule>
<rule>
<!-- Pattern -->
<target-element type = "Condition"/>
<TD>
<eval><![CDATA[
getAttribute("Type")
        ]]></eval>
</TD>
</rule>
</xsl>
```

(b)

```
C>msxsl -i cfs.xml -s cfs4.xsl -o cfs4.htm
C>type cfs4.htm
<HTML>
<HEAD>
<TITLE>Cars for sale - Example 4</TITLE>
</HEAD>
<BODY>
<TABLE BORDER="1">
<TR>
<TD>Number</TD><TD>Price</TD><TD>Maker</TD><TD>Condition</TD>
                                                        <TD>Color</TD></TR>
<TR><TD>1</TD><TD>10000 Dollars</TD><TD>Toyota</TD><TD>Good</TD><TD>Red</TD>
</TR>
<TR><TD>2</TD><TD>20000 Irish Punts</TD><TD>Ford</TD><TD>Good</TD><TD>
White</TD></TR>
</TABLE>
</BODY>
</HTML>
```

Listing Five

(a)

```
C>type cfs5.xsl
<xsl>
<rule>
<root/>
<HTML>
<HEAD>
<TITLE>Cars for sale - Example 5</TITLE>
</HEAD>
<BODY>
<TABLE BORDER="1">
<children/>
</TABLE>
</BODY>
</HTML>
</rule>
<rule>
<target-element type = "Car"/>
<TR>
<select-elements>
<target-element type = "Maker"/>
</select-elements>
</TR>
</rule>
<rule>
<target-element type = "Maker"/>
<TD>
<children/>
</TD>
</rule>
</xsl>
```

(b)

```
C>msxsl -i cfs.xml -s cfs5.xsl -o cfs5.htm
C>type cfs5.htm
<HTML>
<HEAD>
<TITLE>Cars for sale - Example 5</TITLE>
</HEAD>
<BODY>
<TABLE BORDER="1">
<TR>
<TD>
Toyota
</TD>
</TR><TR>
<TD>
Ford
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Listing Six

(a)

```
C>type cfs8.xsl
<xsl>
<define-script><![CDATA[
// 1.5 Dollars to every Irish Pound
var ExchangeRate = 1.5;
// Convert price into Irish Pounds based on the ExchangeRate variable
// if units is Dollars
function getPriceInIrishPunts(price,units)
{
 if (units == "Dollars")
 return price * ExchangeRate + " Irish Pounds";
 else
 return price + " Irish Pounds";
}
]]></define-script>
<rule>
<!-- Pattern -->
<root/>
<!-- Action -->
<HTML>
<HEAD>
<TITLE>Cars for sale - Example 8</TITLE>
</HEAD>
<BODY>
<P><B>
Note: Exchange Rate Used <eval>ExchangeRate+" Dollars per Irish Pound"</eval>
</B></P>
<children/>
</BODY>
</HTML>
</rule>
<rule>
<!-- Pattern -->
<target-element type = "Car"/>
<!-- Action -->
<P>
Price in Irish Punts= <eval><![CDATA[
getPriceInIrishPunts(getAttribute("Price"),getAttribute("Units"))
]]></eval>
</P>
<children/>
</rule>
<rule>
<!-- Pattern -->
<target-element type = "Make"/>
<target-element type = "Color"/>
<!-- Action -->
<P>
<children/>
</P>
</rule>
</xsl>
```

(b)

```
C>msxml -i cfs.xml -s cfs8.xsl -o cfs8.htm
C>type cfs8.htm
<HTML>
<HEAD>
<TITLE>Cars for sale - Example 8</TITLE>
</HEAD>
<BODY>
<P><B> Note : Exchange Rate Used 1.5 Dollars per Irish Pound
</B></P>
<P> Price in Irish Punts= 15000 Irish Pounds
</P>Toyota<P>
Red
</P><P> Price in Irish Punts= 20000 Irish Pounds
</P>Ford<P>
White
</P>
</BODY>
</HTML>
```

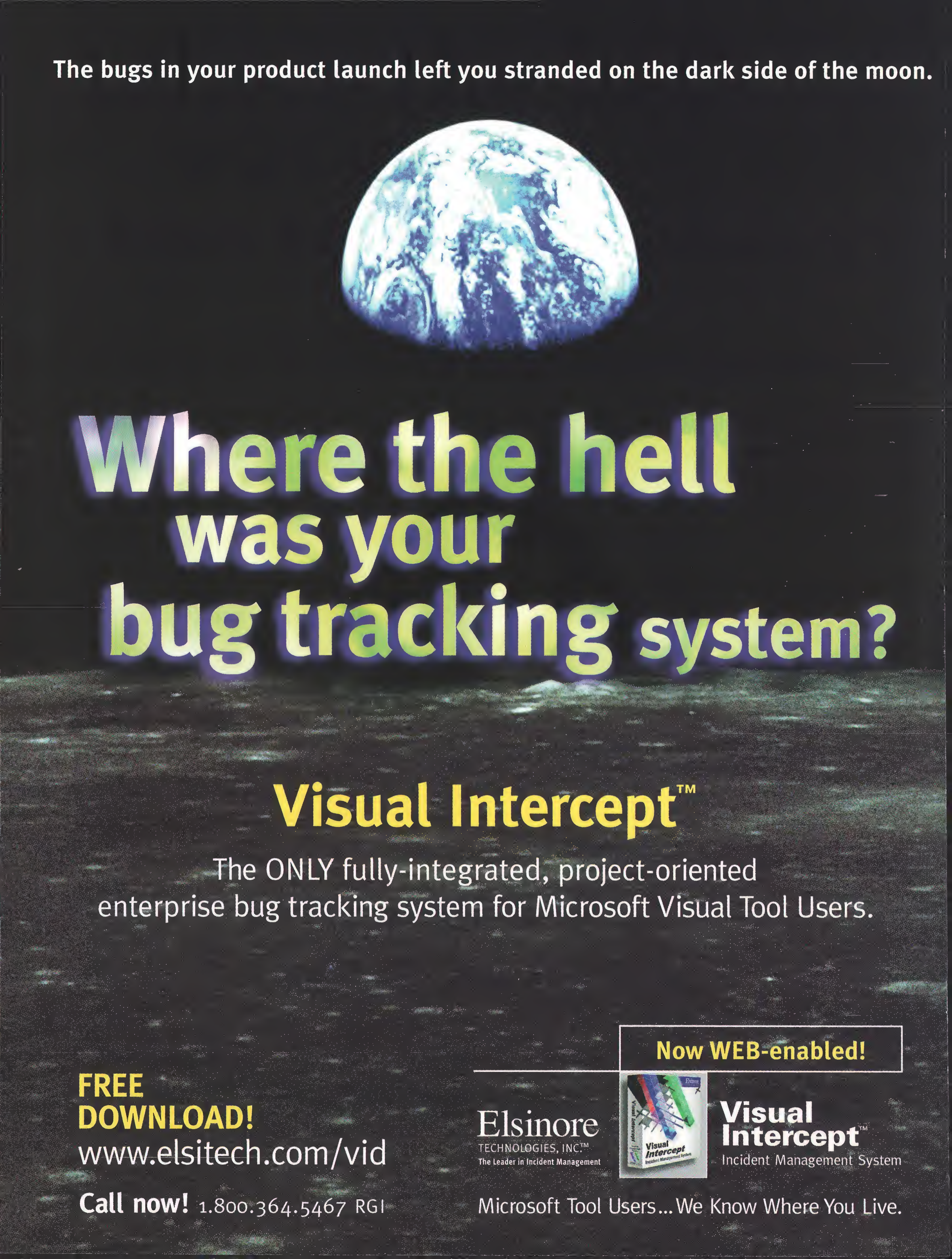
DRAGON

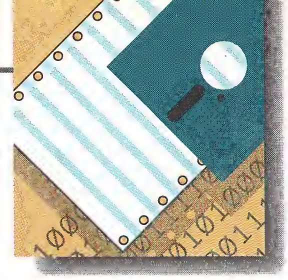
Listing One

```
VERSION 5.00
Object = "{C9F1DD69-49F9-11D0-B5C5-444553540000}#1.0#0"; "dd32.ocx"
Object = "{648A5603-2C6E-101B-82B6-000000000014}#1.1#0"; "MSCOMM32.OCX"
Begin VB.Form MainForm
   Caption         =   "Voice Dialer"
   ClientHeight    =   3195
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   4680
   LinkTopic       =   "Form1"
   ScaleHeight     =   3195
   ScaleWidth      =   4680
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton ManDial
      Caption         =   "Number"
      Height          =   495
      Left            =   120
      TabIndex        =   3
      Top             =   1560
      Width           =   975
   End
   Begin MSCommLib.MSComm MSComm1
      Left            =   720
      Top             =   2520
      _ExtentX        =   1005
      _ExtentY        =   1005
      _Version        =   327680
      DTREnable       =   0   'False
   End
   Begin VB.CommandButton Delete
      Caption         =   "Remove"
      Height          =   495
      Left            =   120
```

```
      TabIndex        =   2
      Top             =   840
      Width           =   975
   End
   Begin VB.CommandButton Add
      Caption         =   "Add"
      Height          =   495
      Left            =   120
      TabIndex        =   1
      Top             =   120
      Width           =   975
   End
   Begin VB.ListBox List1
      Height          =   2790
      Left            =   1320
      Sorted          =   -1  'True
      TabIndex        =   0
      Top             =   120
      Width           =   3135
   End
   Begin DDSpeechLib.DDSpeech DDSpeech1
      Left            =   120
      Top             =   2640
      _Version        =   65536
      _ExtentX        =   741
      _ExtentY        =   741
      _StockProps     =   0
   End
End
Attribute VB_Name = "MainForm"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Private Sub Add_Click()
' Name, number, and generic string
Dim n As String, nm As String, s As String, word As String
AddForm.Show vbModal
If AddForm.Cancelled <> True Then
  n = AddForm.NewName
  nm = AddForm.NewNumber
  s = n & Chr(9) & nm
  List1.AddItem s
  word = s & "[" & n & "]"
  If DDSpeech1.AddWord("PhBook", "TelNum", word, "'")
                                          = EXP_ERR_WORD_HAS_NO_MODEL Then
    DDSpeech1.TrainWord = word
  End If
Unload AddForm
End If
End Sub

' Dial a number in the format of name (tab) number [xxx]
' The brackets, if present at all, are ignored
Sub Dial(ByVal word As String)
Dim n As Integer
Dim t0 As Date
Dim dn As String, nam As String  ' Dial number, name
n = InStr(word, Chr(9))
dn = Right(word, Len(word) - n)
nam = Left(word, n - 1)
n = InStr(dn, "[")
If n <> 0 Then dn = Left(dn, n - 1)
MSComm1.PortOpen = True
MSComm1.Output = "ATV1E0DT" & dn & Chr(13)
t0 = DateAdd("s", 5, Now)
Do
    DoEvents
Loop Until Now > t0  ' Wait 5 seconds
MSComm1.PortOpen = False
MsgBox dn, vbOKOnly, "Dialed " & nam
End Sub

'Delete Entry
Private Sub Delete_Click()
Dim n As Integer
Dim word As String, nam As String
n = List1.ListIndex
If n <> -1 Then
  If MsgBox("Delete this entry", vbYesNo) = vbNo Then Exit Sub
  word = List1.Text
  nam = Left(word, InStr(word, Chr(9)) - 1)
  word = word & "[" & nam & "]"
' Delete word from dragon dictionary
  If DDSpeech1.DeleteWord("PhBook", "TelNum", word) Then
    List1.RemoveItem n
  Else
    MsgBox "Can't remove name"
  End If
Else
  MsgBox "Please select a name first"
End If
End Sub

' Manual dial a number
Private Sub ManDial_Click()
Dim nr As String
nr = InputBox("Enter or say the number to dial")
If nr <> "" Then Dial ("Manual Dial" & Chr(9) & nr)
End Sub

Private Sub DDSpeech1_SpeechRecognized(word As String, WordValue As String)
Dim SearchWord As String
Dim i As Integer
' Find string in listbox so we can highlight it
SearchWord = Left(word, InStr(word, "[") - 1)
List1.ListIndex = -1
For i = 0 To List1.ListCount - 1
  If SearchWord = List1.List(i) Then
    List1.ListIndex = i
    Exit For
  End If
Next i
Dial word  ' Do it
End Sub

Private Sub Form_Load()
Dim s As String
Dim n As Integer
' Start Dragon if not already started
If Not IsDDWinRunning() Then
  If Not StartDDWin() Then
    MsgBox "Can't start Dragon Dictate", vbExclamation
    End
  End If
End If
DDSpeech1.Attach = True
DDSpeech1.AddVocabulary "PhBook"
DDSpeech1.AddGroup "PhBook", "TelNum"
DDSpeech1.Vocabulary = "PhBook"
DDSpeech1.Group = "TelNum"
' Load phone numbers already in vocabulary
s = DDSpeech1.WordFirst
Do While s <> ""
  n = InStr(s, "[")
  List1.AddItem (Left(s, n - 1))
  s = DDSpeech1.WordNext
Loop
End Sub

' Double click for those who are speechless!
Private Sub List1_DblClick()
Dial List1.Text
End Sub
```

DDJ


The bugs in your product launch left you stranded on the dark side of the moon.
Where the hell was your bug tracking system?
Visual Intercept™
The ONLY fully-integrated, project-oriented enterprise bug tracking system for Microsoft Visual Tool Users.
Now WEB-enabled!
FREE DOWNLOAD!
www.elsitech.com/vid
Call now! 1.800.364.5467 RGI
Elsinore
TECHNOLOGIES, INC.™
The Leader in Incident Management
Visual Intercept™
Incident Management System
Microsoft Tool Users... We Know Where You Live.



Leo and the Lizards

Michael Swaine

Embarrassed at never having read *Finnegan's Wake*, I recently tried to struggle through one of the many guides to James Joyce's dense masterpiece. All I got for my efforts, though, was a lamentable penchant for portmanteau words, as in the subhead below. A portmanteau word is an invented word constructed by compressing two or more words into the space reserved for one. "Thissue" compresses "this issue" and "calmtense" compresses "column contents," as well as suggesting a portmanteau mood for this month's column: calm/tense. You know: Relax and enjoy the roller-coaster ride.

Tracking the ups and downs of computer technology, the paradigmatic undulations, the huckster hype and the cotton candy fluff of vaporware, has always felt to me like a day at the carnival. To lessen any vertigo induced by this column's roller-coaster ride, I've instituted the next section, a sort of table of contents for the column. Relax and read on.

Thissue Calmtense

- Last year all new product names sounded like Starbucks varieties. This year there's a small but noisy flurry of new projects that sound like Japanese movie monsters. Mozilla, Cryptozilla, Scandizilla, Rhapzilla, Raptor. Netscape has let the lizards loose.
- There's this program called LEO. It combines Knuth-style Literate Programming with outlines à la Baron. Could an old Friend of Dobb's have improved on Knuth?
- Wasn't the Fifth Generation Project a pop group back in the '70s? Not exactly.
- Quantum computing wakes up in chlorophyll.
- There's this computer called Leo. It may have been the first business computer.
- In Paradigms Past: Who actually invented the calculator? Please tell me he wasn't called Leo.

Land of the Lizards

This spring, Netscape did what it had promised and made the source code available for its Communicator product, inviting the entire software-development community to join in the process of developing the next version of Communicator, and, not incidentally, any products of their own they'd like to spin off. Within two weeks, 100,000 people had downloaded the code.

Some of the code was missing, for the very good reason that Netscape didn't control the copyright on software it had licensed from others. Some was missing for a dumb reason: the U.S. government policy on encryption software. Within seven hours, the missing crypto had been replaced, and the U.S. government made a monkey of, by Australian programmers.

Days later, James Clark, the technical lead on the World Wide Web Consortium's XML working group, (no relation to Netscape cofounder Jim Clark) added his XML parser to the Mozilla code base, and Netscape released the source for JavaScript 1.3. And so on. Collaborative programming was off and running.

Netscape likes to call this Open Source software. But Netscape didn't invent it (and, to be fair, doesn't pretend otherwise). It was called Free Software when Richard Stallman seemed to be all alone out there promoting the spirit of sharing what he learned at MIT in the '60s. Stallman's crusade always seemed noble but quixotic. Sure, it would be a nicer world if everyone shared their programming discoveries with their programming peers. Sure, technology would progress more rapidly if programmers didn't have to invent the virtual wheel every day because some earlier wheel inventor had locked up the plans. Sure, we'd all stand taller if we could stand on the shoulders of giants. But software development is (often) a business, and intellectual property has dollar value.

And so Stallman and the GNU heroes labored almost unknown. Until the Web arrived.

Maybe it was just because the suits hadn't figured out yet how to make money on the Web, maybe it was because the Web got its start among academics, but for some reason the tools of choice among webmasters were often free tools. Linux, Perl, Apache. The Apache Web server's name even suggested the collaborative development process that Netscape is encouraging— a patchy construction, a quilt. And Netscape's cofounder was both savvy enough to understand the virtues of the approach and gutsy enough to push his company to put on a quilting bee.

Now, thanks to Netscape and Marc Andreessen, the Free Software/Open Source movement has momentum. Let's hope it continues. One place to watch or join the movement is http://www.mozilla.org/, but there's also http://www.opensource.org/.

The Tangled WEB

I have recently been looking at a source code editor whose author provides the source code with the product. Although I think that Ed planned to release the source for LEO long before Netscape did ditto for Communicator.

LEO was inspired by Donald Knuth's model of literate programming. Its inventor, Ed Ream, who also invented RED (which long-time *DDJ* readers will recognize as a text editor published in *DDJ* some 15 years ago), has been fiddling with literate programming for over a decade, but it never quite worked for him.

Literate programming came onto the scene about the time Ed wrote RED. Knuth published the first paper on literate programming, and the language-plus-programs that embodied it, back in 1984. Looking for a three-letter English word that hadn't already been applied to computers, Knuth decided to call his literate programming system WEB. (Later CWEB, but I'll use the earlier term as a generic here.)

When you write a program in WEB, you break your code down into sections, and these sections, written in C or Pascal or whatever plus WEB syntax, serve as the source code for two different WEB routines. One produces documentation that describes the program clearly and facilitates debugging. Its output goes to a text-formatting program such as Knuth's own TEX. The other produces the machine-executable code, which serves as input to a compiler or interpreter. Since the same source generates both the documentation and the executable code, they are sure to be consistent with one another. The best introduction to literate programming is the book of the same title, by Donald E. Knuth (*CSLI Lecture Notes No. 27*, 1992; ISBN 0-937073-80-6).

Ed's problem with WEB was that it was too hard to know when to create new sections and to keep track of all the sections

---

Michael is editor-at-large for DDJ. *He can be contacted at mswaine@swaine.com.*

Paradigms Past

It seems like a simple question: Who invented the calculator? But questions of paternity are often tricky. According to the U.S. Patent Office, the inventor was a bank clerk in St. Louis named William Seward Burroughs, in 1886. Burroughs, the namesake and ancestor of beat author William S. Burroughs, built both a calculator and a company to sell it (American Arithmometer Company, later Burroughs Adding Machine Co.).

But Burroughs was beaten to the punch (by fully 66 years) by one Charles Xavier Thomas, Thomas of Colmar to his friends, who built the first commercial mass-produced calculator in 1820. It could add, subtract, and multiply, and, if you helped it a little, even divide. It took up most of a desktop, and continued to be sold for 90 years.

Tom had got the idea, though, from a 17th century invention of Gottfried Wilhelm von Leibniz. Leibniz's Stepped Reckoner was definitely a calculator: It added, subtracted, and did multiplication by repeated addition and shifting. Although Leibniz was an early booster of the binary system, his machine was decimal.

But Leibniz wasn't first: Three years before he even planned his machine, a Brit named Samuel Morland had built a machine for toting up (decidedly nondecimal) British currency.

Was Morland the first? Nope. Both he and Leibniz had merely expanded on an invention of Blaise Pascal. The Pascaline, built in 1642 for Pascal's tax-collector dad, was (aha!) the first digital adding machine (and the first digital business machine). Pascal sold about a dozen of them. But the story doesn't end there. Still earlier, in 1624, a fellow named Schickard had built a Calculating Clock that could add and subtract. If you really needed to multiply, you could use the slide rule affixed to the front.

And even earlier, sometime in the 1500s, an artist drew some (recently discovered) sketches for a mechanical device that would add and subtract numbers. When a machine was built based on these sketches, it actually worked.

The name of this artist, arguably the true inventor of the calculator, was Leonardo da Vinci. Another Leo.

—M.S.

once they were created. He now says that "flat literate programs have too little structure." His solution: Add outlines.

There's a devilish irony in Ed's decision. One reason that Knuth chose the name WEB for his system was that he was coming to realize that programs are better understood as webs than as hierarchical structures. He wanted to get away from top-down, hierarchical programming models. WEB was designed to allow programmers to write top-down, bottom-up, some-of-each, or stream-of-consciousness, and still be able to convey to a human reader the relationships among the parts of the program. Ed put back the hierarchy, using More, the outliner from Userland written by Doug Baron et al., as his model.

What we're talking about here: An outline consists of headlines that can be moved simply by dragging them. Each headline contains body text. Headlines define the organization of the document; a parent outline can contain zero or more children. Children are indented from their parents. You can expand or contract headlines simply by double-clicking near the headline. (Yes, double-clicking. Ed is currently developing on a Mac, although his real target platform is Rhapsody.) Ed soon decided that outlines solved all his problems with literate programming. Since then he's been doing all his programming using Leo and literate outlines, and says he'd never willingly program in any other form.

Knuth has said the same thing about *his* approach, but, as Ed points out, the programming world has not beaten a path to Knuth's door demanding WEB. Maybe it lacks something. Maybe that something is outlines.

LEO is specifically a text editor for C programs (it also supports C++ and Objective-C syntax). It employs a simple language called SWEB, based on Knuth's WEB, but considerably simpler (Ed took out all the typesetting code). Your program is expressed primarily in the body text of headlines that you create. This body text consists of plain C (C++, Objective-C) code, with additional directives defined by the SWEB language. You use LEO's Tangle command to translate the SWEB code to C code.

The combination of outlines and SWEB, Ed claims, makes programming significantly easier and more fun. And while outlines impose a hierarchical structure, they don't impose a *single* hierarchy. You can organize and reorganize your program at will, creating different views and organizations of the same program.

Leo in Action

In LEO or in WEB, you use this notion of sections to structure your code. Ed claims several virtues for sections:

- A section name is more descriptive than a function name, and helps create self-documenting code.
- Sections provides a convenient place for lengthy comments that would otherwise clutter up code.
- Sections can be used to define nonexecutable code such as data structures.
- Defining sections is easier than defining macros or functions; there is never any need to create prototypes for sections.
- Sections may access the local variables present where the section is referenced, just as in a macro expansion. Usually this is exactly what is wanted, and there is no need to define and pass parameters.

As I said, Ed is making the source to LEO available (not for free, though, and subject to a license that I haven't seen yet). Example 1 is a snippet of the code for the *print* function, written using LEO, to demonstrate that coding with LEO isn't too unfamiliar.

Up, Up, and Away

I was wondering whatever became of the Japanese Fifth Generation Computing effort, so I did some research.

In 1981 Kazuhiro Fuchi announced the Fifth Generation Project, the same year U.S. artificial-intelligence pioneer Edward Feigenbaum started the first successful artificial-intelligence company, Teknowledge. The following April, the Fifth Generation Project was officially launched with a multimillion-Yen budget to develop hardware and software, focusing on parallel processing and logic, to solve problems requiring inference.

Hundreds of researchers worked on the project at any one time, and it became one of the best training grounds for computer scientists in Japan. Five different computers came out of the project over the next ten years, as well as a specialized operating system, database systems, and programming tools. In 1992, at the end of the project, Feigenbaum visited Japan and judged the Japanese work to be at least on a level with work done independently in the U.S. Some of the work was definitely the most advanced in the world. But traditional PCs and workstations soon surpassed the power of the Fifth Generation hardware. And the software, written for the obsolete hardware, was largely ignored. In 1992, The Ministry of International Trade and Industry granted the researchers a two-year extension to port the software to UNIX, and the Fifth Generation Project dissolved in 1995.

The software, which was placed in the public domain, may yet be used on UNIX-based highly parallel multiprocessor

machines, but the most important outcome of the Fifth Generation Project may be that it gave an unparalleled education to a generation of young computer scientists.

Two Bits Make a Quantum Leap

Man was not meant truly to understand quantum physics. Not this man, anyway. I read a lot about it, and just when I think I've really understood some crucial concept, I look at it from another point of view and it makes no sense to me.

When quantum computing takes over, I'm going to be in trouble. Fortunately, it looks like quantum computers won't bump the PC off my desk or yours. They will be used, when they are actually practical, for certain specialized tasks, such as factoring large numbers. Okay, some of you are thinking of reasons why you might want to factor large numbers on your desktop PCs. Maybe you'll have QCs alongside or instead of your PCs.

By encoding information in spin states of a proton, say, which exist as a superposition of both 0 and 1 until a measurement is made, a QC can theoretically explore different paths through a mathematical problem simultaneously, making them useful for factoring and cryptographic work.

But quantum computers are a long way from practical today. It's amazing enough that they are even considered possible. Now Isaac Chuang of IBM's Almaden Research Center and Neil Gershenfeld of MIT are claiming that they've built one and that it can answer two questions about four numbers, like "which of the number 1, 2, 3, or 4 is greater than 2 and odd." The "computer" consists of the nuclei of a carbon atom and a hydrogen atom in a chloroform molecule, manipulated by magnetic fields and radio waves.

According to Lov Grover, a physicist at AT&T Bell Labs, Chuang and Gershenfeld have demonstrated "that quantum computing works, not just with pencil and paper, but in the lab."

LEO Gets Down to Business

The first business computer was not a quantum computer but a cold-cut computer. It was built by a British catering company, according to a new book by David Caminer, John Aris, Peter Hermon, and Frank Land (*LEO: The Incredible Story of the World's First Business Computer*, McGraw Hill, 1997).

If that's not weird enough for you, how about this: In 1947, two bean counters from Lyons & Co., the catering company, visited the Princeton lab where ENIAC was being built. They went home and told management, in effect, "We oughtta get us one of these things." Lyons funded some computer research at Cambridge, where some of the fundamental steps in inventing the digital computer were being taken, and a couple of years later took the fruits of that investment, hired some engineers, and built a computer. They were up and running with custom catering-business software by 1951.

The computer was called the Lyons Electronic Office, or LEO. Then again, establishing firsts in computer history is like defining nationhood. A fictional Irishman once defined a nation as "the same people living in the same place. Or also living in different places." That was Leopold Bloom, from James Joyce's *Ulysses*, and another Leo.

DDJ

```
void print(void)
{
 FTAG("print");
 <<< Define print vars >>>
 STATB(ftag);

 <<< Return if there is nothing to print >>>
 <<< Save the old port >>>
 PrOpen(); if (PrError() != noErr) goto done;
 <<< Create a TPrint Record.  goto close on error >>>
 cur_res_file = CurResFile();
 PrValidate(print_h);
 if (PrJobDialog(print_h) == FALSE) {
  cancel_flag = TRUE; goto close;
 }
 <<< Initialize the page counts >>>
      ...
 <<< Warn about any print errors >>>
done:
 <<< Restore the old port >>>
 STATX(ftag);
}
```

Example 1: *Sample LEO code.*

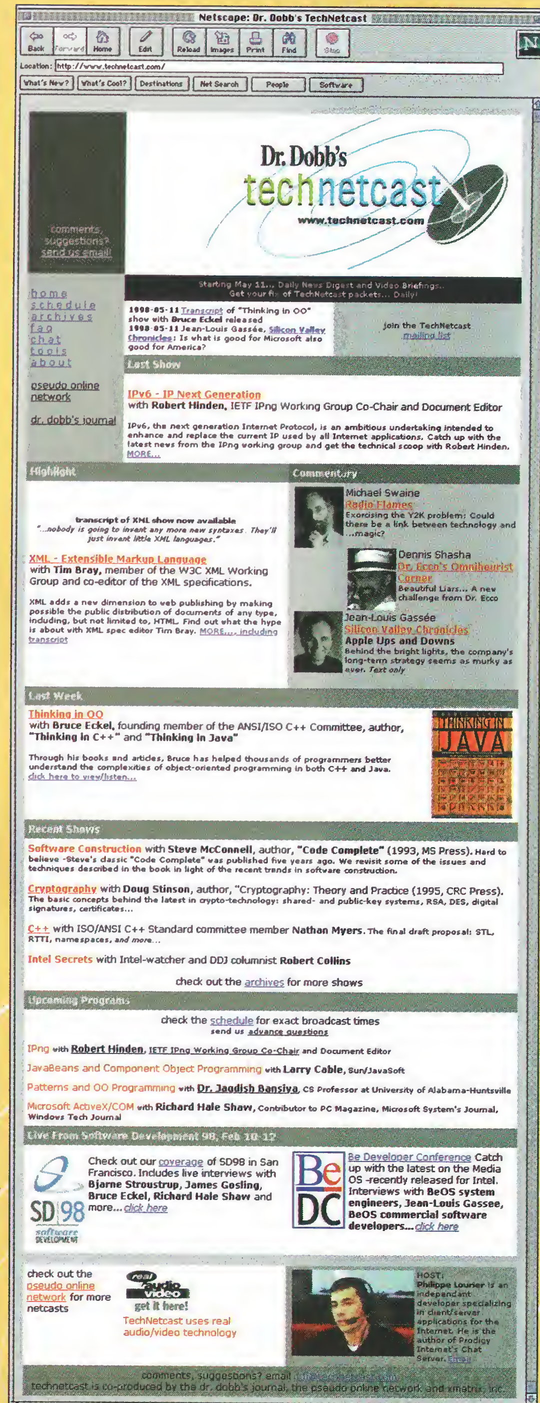
DR. DOBB'S

TECHNETCAST

audio+video+chat events for developers

Announcing TechNetcast Daily Briefings... Programmer news, opinions, reviews... A high-volume burst of TechNetcast packets... daily!

www.ddj.com



News

Visit technetcast.com for daily news digests and video briefings. Specifications, product releases, events... the latest technology news for programmers.

Upcoming Programs: send us advance questions; ddj@technetcast.com

Mozilla.org. Open Software . . . . with Tom Paquin, mozilla.org

Graphics Programming: Reactive Animations . . . . with Conal Elliott, ... Mircrosoft Research

CORBA . . . . with Andreas Vogel, Visigenics Software

MS DCOM . . . . with Roger Sessions, ObjectWatch Inc.

And live coverage from Win-Dev East 98 and Web 98
Check Web site for broadcast dates

Regular AudioColumns

Radio Flames by Michael Swaine . . . . **bi-weekly, audio + text**

Silicon Valley Chronicles by Jean-Louis Gassée . . . . **weekly, text**

Dr. Ecco's Omniheuristic Corner by Dennis Shasha . . . . **monthly, audio + text**

Archives

JavaBeans and Component Object Programming with Larry Cable
Component object programming is changing the way software is written. JavaSoft's Larry Cable presents the latest in Java component technology.

IPv6, IP Next Generation with Bob Hinden
Catch up with the latest news and get the technical details about the next generation Internet Protocol with IPng WG Co-Chair Bob Hinden

XML, Extensible Markup Language with XML WG Editor Tim Bray
XML adds a new dimension to web publishing by making possible the public distribution of documents of any type, including, but not limited to, HTML. Find out what the hype is about and get the technical scoop with XML spec editor Tim Bray.

Thinking in OO with Bruce Eckel
Through his books and articles, Bruce has helped thousands of programmers better understand the complexities of object-oriented programming in both C++ and Java.

Special Events

Learn about BeOS development with our coverage of BeDC. Includes interviews with Jean-Louis Gassée, the BeOS engineering team (BeOS system internals) and Be application developers (developing commercial Be applications).

host: Philippe Lourier
email: ddj@technetcast.com
www.technetcast.com

TechNetcast is produced by *Dr. Dobb's Journal* and the Pseudo Online Network (www.pseudo.com)

Windows CE

Al Stevens

Last month, I described my first foray into Windows CE programming. I talked about the usability and performance of the first generation handheld PC (HPC) devices. I also rued the limited ability of this platform to support real-time applications. Since writing that column, I attended the Windows CE Developer's Conference. This conference is in its fourth year and is hosted by Microsoft. Attendees are typically C and C++ programmers who are or want to be developing programs to run on Windows CE devices. The new SDK also supports CE application development with Visual Basic.

Microsoft now identifies three specific platforms for Windows CE devices, each with its own unique programming considerations.

- First are the so-called PC Companion devices, which include HPCs and the new breed of Palm PCs. They're called PC Companions, because you need a desktop or laptop PC to print, archive, and transmit your information. Companions have a screen and a GUI with a Start Menu and all that. HPC devices have keyboards. Palm PC companions do not.
- The second platform is the Auto PC, a specialized device for your car's dashboard.
- Third is the embedded systems platform, which might not have a screen or a keyboard or any other device other than memory and a microprocessor and which probably does have proprietary devices with custom device drivers written by the product developers.

The Windows CE SDK plug-in to Visual C++ supports all these platforms.

If you are a big fan of Windows CE, you might not like what I have to say. In my considered opinion, Windows CE in its current incarnation isn't there yet. Its three platforms, cute and endearing as they are, fall short of adequately supporting their respective targeted users for different reasons, which I'll address later. Its development environment—the Windows CE SDK running under Windows NT 4.0—is buggy, fragile, and temperamental, as you too will be after trying to install it.

Those opinions notwithstanding, you are likely to find gainful employment as a C++ programmer who knows the Windows CE development environment because Microsoft is in full promotional hyperspeed and, if the growing attendance at this conference is any indication, product managers and their developers are listening.

Al is a DDJ *contributing editor. He can be contacted at astevens@ddj.com.*

New HPCs

HPCs have an allure, to be sure. The other day, I was looking at a display of handheld devices at Computer City. A fellow standing next to me was slathering over a first generation Casiopeia A-11, now reduced to half its original price. I told him I have one and hate it and said why. He pointed to a PalmPilot and said he couldn't understand why anyone would want one of those little devices. I answered that PalmPilot has about a million happy users. He ignored all that valuable witness, turned his attention back to the A-11, and with a glazed stare wondered out loud if his company would buy him one. Logic cannot penetrate the protective shields of an enamored consumer about to part with plastic.

All the vendors who exhibited at the conference displayed new HPCs with Windows CE 2.0. Each new machine sports a slightly bigger keyboard than the first generation, and several have well-lit color displays. Those with monochrome displays, even with their backlighting, continue to be unreadable in all but the brightest of lighting conditions. Only two vendors, NEC and Hitachi, have keyboards that I would consider using to touch-type. These machines are pricey but have some potential for road warriors. They are not pocket rockets, being just a bit smaller than the smallest notebook computers. They use proprietary battery packs, which sacrifices the convenience of flashlight batteries and adds a charger, power cord, and maybe a second battery pack to what you have to lug around. They are companions; you still need a desktop connection. In other words, new generation HPCs have few of the advantages of the original HPCs, or of conventional notebooks, and most the disadvantages of both. The form factors are growing, but the technology has yet to reach maturity.

Want Your Palm Read?

During the conference, we learned that 3Com, purveyors of the very popular PalmPilot product line, had sued Microsoft in Europe for trademark infringement over the use of the name "Palm PC." We then heard that 3Com had prevailed in that action, forcing Microsoft to find a different name for the line, which several manufacturers have already begun to produce. Microsoft did not comment on this judgment, but, without enough time to change its slide shows, continued to use the name "Palm PC" to represent those products throughout the conference. After the conference, we learned that the new Microsoft moniker would henceforth be "Palm-size PC."

Several booths allowed attendees to try out Palm-size PC devices. My first reaction: Microsoft doesn't get it. The PalmPilot is popular because it is simple and easy for users and programmers. PalmPilots do not have multitasking, Control Panels, Registries, drivers, and exotic Setup programs to confound programmers and confuse users. Palm-size PC devices, on the other hand, have all that. They are narrow Windows CE devices without keyboards. The Windows UI paradigm, complex enough on desktops and marginal at best on HPCs, doesn't port well at all to the smaller Palm-size PC form factor. (That's twice. "Form factor" is a new buzzphrase to add to your technocabulary. Every pitch at the conference was liberally sprinkled with form factors.)

My reaction to the Palm-size PC? All conference attendees were promised one to be mailed in a month or two when the little boxes are in full production. I'll reserve final judgment until I get mine (assuming

that lowly members of the press qualify for the freebie; I'm sure not going to buy one). At first glance, the product line looks like a knee jerk reaction to someone (3Com) who is making a lot of money selling a computing device that Microsoft does not control.

Auto PC

The second Windows CE platform is the Auto PC, which is a device to be installed in the dashboard of a car. It is voice actuated and presumably can send and receive e-mail, trade stocks, and do other urgent tasks that cannot wait until you get where you are going. It also reports on your car's status—something that gauges and idiot lights apparently no longer do well enough—and connects to a GPS so you don't get lost on your way to the local cyber cafe. There were three luxury cars parked on the conference tradeshow floor. Attendees could climb in and try out the Auto PCs while spectators viewed the devices on video monitors. Every time I went to look at one, somebody was sitting in the driver's seat, yelling at the device, trying to get it to do something. I overheard a young programmer express his enthusiasm about the Auto PC after climbing out of the Jaguar convertible. "What a cool device! I've got to have one! As soon as they are available, I'm getting one, for sure!" Then, after a pause he added, "I guess I'll have to get a car, too."

The voice-recognition Auto PC opens a whole new arena for road rage. Instead of flipping the bird or shooting out someone's tires, you can pull up next to their convertible and yell at their Auto PC, "Buy 10,000 shares of Borland!" or "Forward all personal e-mail to wife!"

I give Auto PCs the same chance for universal acceptance that pen computing enjoyed a few years ago. Maybe Generation X or whoever follows them into the 21st century will find compelling uses for this new technology while it waits for some important problem to solve. Remember CB radio? I'm hanging on to my road maps and watching the mechanical fuel gauge at least for a while. This skepticism kept me from going to any of the Auto PC technical sessions, so I have nothing more to report. I just don't see any future in it.

Embedded Systems

Microsoft is positioning Windows CE as an embedded operating-system platform. The company presents this position as if it was the plan all along, but I smell revisionism. Every session that addressed the Embedded Developer's Kit (EDK), or embedded development in general, paid lip service to real time requirements. It grudgingly acknowledges that Windows CE, which uses the Windows operating model for sending notification messages to applications in response to events, is not a real-time operating system. Windows CE ISRs cannot be nested; an event cannot interrupt an interrupt service routine. They suggest that a program should signal a thread from its ISR and exit the ISR as soon as possible to permit other interrupts to be sensed and processed. The mechanics of a thread receiving a signal involves the Windows messaging system, which sends a notification message to a window whenever the heck it feels like it. Windows CE 2.0, the current version, does not correct this problem; neither does 2.1, soon to be released. Microsoft expressed its commitment to solving the real-time issue with interrupt and ISR latencies of 50 and 100 microseconds in a future version; they seemed surprised that a significant majority of the attendees were embedded system programmers.

This apparent real-time support deficiency in Windows CE prompted many offline discussions. At lunch one afternoon, one of the guys polled for a consensus definition of "real time." Those at the table generally agreed that real-time requirements are a function of the application and can be expressed in measurements that range from nanoseconds to fortnights. How fast must a system respond to an external event and what are the consequences if the response is late or lost? Is the user taking an online examination, trading stocks, piloting a spacecraft, or stitching a microscopic suture during bypass surgery? From that consensus we agreed that Windows CE could claim to be a real-time operating system only for applications with real-time requirements that fit within CE's operating envelope.

In other words, you can't define real time outside of the context of some specific real-time requirements. Given that vague definition, a batch payroll system could be called real time if it consistently delivers paychecks to the employees by quitting time on payday. Which is why I reject that definition.

Until Microsoft adds true real-time support, it cannot credibly claim that Windows CE is a serious player in the embedded operating-system market.

C++ Exception Handling: Why Not?

During several technical sessions, the presenters pointed out that Visual C++ does not support Standard C++ exception handling (the try/throw/catch idiom) when compiling to a Windows CE target. They

made vague references to platform incompatibilities. That reason does not make sense to me. C++ exception handling is a language issue implemented in code. Its implementation has nothing to do with the platform. The compiler emits code into statement blocks that can throw exceptions. That code stores in a table somewhere the addresses of automatic instances of classes that have destructors and the addresses of the destructors. The throw mechanism uses that table to call the destructors as it unwinds the stack. It's just code that occupies memory.

The only valid reason I can see for disabling the feature is because of the added overhead. I surmised that a platform without enough memory to support exception handling isn't going to be able to do much recursion or store and pass any sizable instance variables. In other words, it isn't going to be able to support languages such as C and C++ that use the stack for local variables and function arguments.

I stopped by the Metrowerks booth to look at the CodeWarrior compiler for Windows CE. They told me the same story. No C++ exception handling or RTTI because of platform restrictions. Figuring I'd have a better chance of talking to a compiler builder at Metrowerks, I called the company. It turns out that Metrowerks fully intends to include exception handling support in a future version of its CE C++ compiler. Metrowerks just hadn't done it yet. But it has successfully implemented the feature in the Metrowerks PalmPilot compiler, which has a much smaller (ahem) form factor than the typical CE device.

PalmPilot

My latest electronic toy acquisition is the 3Com PalmPilot. I got one because it was on sale, because it actually fits in a shirt pocket, because everyone I know who has one likes it, and because there is a C++ compiler for developing applications. This is one delightful little machine. Its elegance is found in its simplicity. I'll report more when I have the Metrowerks compiler and can build an application.

The Windows CE SDK

Last month, I told of my frustrations getting the SDK installed and working in a new NT system. The conference materials included the latest SDK version, one that supports 2.0 and the new platforms.

Since last month, I had learned that Visual C++ and Internet Explorer 4.0 are not compatible on NT 4.0. When both are installed and you run Developer Studio, something happens internally that screws up the Registry irrecoverably. The symptom shows up when you try to log in as other than an administrator. Things stop working and go downhill from there until they reach bottom and NT won't boot. Because of that, I had to reformat the NTFS drive, reinstall NT, and start over again.

I installed VC++ 5.0 and the new CE SDK on my clean NT workstation. Following the instructions, I reinstalled Service Pack 3. Something in this procedure caused NT's Remote Access Server to refuse to start. I removed and reinstalled Remote Access Server. All attempts to access the CE device from this point forward forced NT into an unstable condition in need of a reboot.

Those of you who have had this experience or similar ones are nodding gravely and feeling my pain. Those of you who had no such problems are gnashing your teeth and preparing to fire off e-mail to tell me what an idiot I am. I don't know which group is the majority, but this I do know: Until Microsoft makes this development environment more stable, programmers are going to look elsewhere for a way to write CE programs— or for better platforms to support.

I do not have a happy ending here. I am overdue on this deadline and the problem remains unsolved while I write. Surely it can be solved, and certainly I will solve it, even if I have to call in my secret weapon (my daughter Sharon) who is a Microsoft Certified Systems Engineer on the NT platforms. I'll let you know.

DDJ

How Do I Process Images with Java?

Aaron Michael Cohen

The Java API defines an abstract imaging model that can be used to display and manipulate both static images and sequences of images. The imaging model is defined in terms of the interactions between an abstract class, *Image*, and three interfaces: *ImageProducer*, *ImageConsumer*, and *ImageObserver*. These interfaces and their support classes are defined in the java.awt.image package.

The *Image* abstract class represents a platform-independent displayable image. Images can be created from image files loaded from the local file system or over a network using *getImage(URL url)* or *getImage(URL url, String name)*. These functions will only succeed with files using one of the supported file formats. Currently, JPEG and GIF file formats are universally supported.

Images can also be created with a call to the *createImage()* function of an AWT Component or available Toolkit object. There are several forms of the *createImage()* function:

- *Image createImage(ImageProducer source)* creates an image from an object implementing the *ImageProducer* interface.
- *Image createImage(int width, int height)* creates a blank image of a given size that can be drawn on using a *Graphics* object.
- *Image createImage(byte[] imagedata)* creates an image from JPEG or GIF format data stored in an array. Only available in Java 1.1 or later.
- *Image createImage(byte[] imagedata, int offset, int length)* creates an image from JPEG or GIF format data stored in *length* bytes of an array starting at the given offset. Only available in Java 1.1 or later.

One of the *createImage()* methods lets you create an *Image* from an *ImageProducer*. You can retrieve the *ImageProducer* associated with an image by using the *ImageProducer getSource()* method of the *Image* class.

Drawing onto an image is accomplished by using the methods of the *java.awt.Graphics* class. Create a *Graphics* object associated with the image by using the *getGraphics()* method. Anyone who has programmed in Java even a little has written the code to draw an image into a existing *Graphics* context by calling *drawImage()* inside a *Component's paint()* handler, so I will not cover that in more detail here.

An image also has properties such as width and height, which can be retrieved with methods defined in the *Image* base class. Java has the built-in capability to download and display images in the background. Because of this it is possible to call methods on an *Image* object before the necessary data is available. These methods require an *ImageObserver* as a parameter. For example, the *getWidth()* and *getHeight()* methods each require an object implementing the *ImageObserver* interface as a parameter. When the requested information cannot be returned immediately, these functions return –1.

The *ImageObserver* interface defines one method, *imageUpdate* (see Example 1). When an operation on an *Image* object cannot be completed immediately because the data is not yet available, a thread is created, which loads the data in the background. As the data is loaded, *imageUpdate()* of all registered image observers is called notifying the observer of progress. The *img* parameter refers to the image for which there is new information. The *infoflags* parameter is a set of flags that define what type of information is now available. The meaning of the rest of the parameters is dependent upon the content of *infoflags*. Returning *true* from *imageUpdate()* requests further information on the image. The function should return *false* if the *ImageObserver* is not interested in any further callbacks for this image. All AWT components implement the *ImageObserver* interface. The default behavior repaints the image when additional pixel data arrives. You usually do not have to override the default component implementation.

So where does the image data come from? Each *Image* is associated with an *ImageProducer*, which can be retrieved by calling the *getSource()* method on the *Image* object. The *ImageProducer* is responsible for delivering the image data upon request to *ImageConsumer* objects. An *ImageProducer* object makes calls on the methods of the *ImageConsumer* interface to inform the *ImageConsumer* of the image type, size, and pixel data. Both the *ImageProducer* and *ImageConsumer* interfaces are defined in the java.awt.image package. The relationship between the classes in the Java imaging model looks like Figure 1.

All of the methods of the *ImageProducer* class have to do with *ImageConsumer* objects registering as consumers with the producer object and requesting data. These methods are fairly straightforward:

- *void addConsumer(ImageConsumer ic)* adds a consumer to the *ImageProducer*. The consumer will be delivered image data the next time new data is available.
- *boolean isConsumer(ImageConsumer ic)* determines whether a consumer is currently registered with the *ImageProducer*.
- *void removeConsumer(ImageConsumer ic)* removes the given consumer from the producer's list of consumers.
- *void requestTopDownLeftRightResend(ImageConsumer ic)* requests the producer to resend its image data in top-down, left-to-right order. The producer may choose not to honor this request.
- *void startProduction(ImageConsumer ic)* registers a consumer with the producer and deliver the current image data to the consumer as soon as possible.

More interesting than the methods of the *ImageProducer* interface are the methods of the *ImageConsumer* interface, which an *ImageProducer* calls to deliver the image type, size, and pixel data to the consumer object. These methods can be described as follows:

- *void setDimensions(int width, int height)* is called to notify the consumer of the image's size. This notification will be called before the first call to *setPixels()*.
- *void setColorModel(ColorModel cm)* is called with a *ColorModel* parameter,

Aaron, who is a staff engineer at Intel where he develops video teleconferencing systems, is also the coauthor of Win32 Multithreaded Programming *(O'Reilly & Associates, 1997). He can be contacted at alcohen@ix.netcom.com, or http://www.netcom.com/~alcohen/.*

which will be the color model of most, but not necessarily all, of the pixels delivered to the *setPixels()* method. This notification is optional. If called, it will be invoked before the first call to *setPixels()*.

- *void setHints(int hintflags)* notifies the consumer of the order in which pixels will arrive, which the filter may use to optimize some operations. For example, the COMPLETESCANLINES flag will be set if each call to *setPixels()* will deliver entire unbroken scan lines (rows) of pixels, and the TOPDOWNLEFTRIGHT will be set if the pixels will be delivered in top to bottom, left to right order. This notification is optional. If called it will be before the first call to *setPixels()*.
- *void setProperties(Hashtable props)* is used by the consumer to add some programmer-defined information to the image stream. This notification will be called before the first call to *setPixels()*.
- *void setPixels()* passes the image pixels to the consumer by calling this method one or more times. Each call will deliver a rectangle of image pixels to the consumer. The layout of the pixels delivered will correspond to any information sent in an earlier call to *setHints()*. There are two versions of this method. One which receives 32-bit pixel data and another which receives eight-bit pixel data.
- *void imageComplete(int status)* is called with a status parameter to inform the consumer that a complete static image has been delivered, one frame of a multiframe image has been delivered, or an error has occurred. This notification will be called after the last call to *setPixels()*.

The *setPixels()* method requires more explanation. The full function prototypes of the two *setPixels()* functions look like Example 2. A portion of the image is delivered to an *ImageConsumer* with each call to *setPixels()*, which may be as large as the entire image, or as small as one pixel. The portion delivered in each call to *setPixels()* is a rectangle whose upper-left corner is at *x,y* and has a width of *w* and height of *h*. The data is passed in the array object pixels, which has valid data starting at position *offset*, and *scansize* elements between each row. What all this

```
public boolean imageUpdate(
    Image img, int infoflags,
    int x, int y, int width,
    int height);
```

Example 1: *The* ImageObserver *interface.*

boils down to is that the sample in the pixels buffer which corresponds to the image *pixel(m,n)* is at array index $offset + (n-y)*scansize + (m-x)$. The only valid pixel array sample values are those that lie in the rectangle described by *w*, *h*, *offset*, and *scansize*. The producer will continue to call *setPixels()* until all of the image pixels have been delivered, or an error occurs.

ColorModels

The delivered image data may consist of 32- or 8-bit samples. A *ColorModel* is passed along with the image data in each call to *setPixels()* so that consumers can retrieve and manipulate the color information in the samples. The data in the pixels buffer is defined in terms of the *ColorModel* passed with the pixels. The *ColorModel* passed in the *setColorModel* call is only a hint to the consumer that the majority of pixels will use that model; you still need to check the *ColorModel* passed in each call to *setPixels()*.

The *ColorModel* abstract class defines a set of functions that an *ImageConsumer* uses to map a pixel value to red, green, blue, and alpha values. The functions *int getRed(int pixel)*, *int getGreen(int pixel)*, *int getBlue(int pixel)*, and *int getAlpha(int pixel)* return the components for the given pixel value. Each color component can take on a value between 0 and 255, inclusive, with 0 being the minimum of a component and 255 being the maximum. The alpha component quantifies the transparency of the pixel and ranges from 0, which lets an underlying image show through completely, to 255, which is opaque.

All *ColorModel* objects also define a function, *int getRGB(int pixel)*, which returns the color of a given pixel value in terms of the default color model, which is known as the *RGBDefault* color model. An instance of the *RGBDefault* color model is returned by calling the *ColorModel* static function, *getRGBdefault()*. The *RGBDefault* color model uses 32-bit data samples and allocates eight bits each for red, green, blue, and alpha with a bit layout of 0xAARRGGBB.

A *ColorModel* object that allocates a portion of the total bits in a sample to each color component is known as a *DirectColorModel*. A color model which uses the pixel value as an index into a palette of colors is called an *IndexedColorModel*. *DirectColorModel* objects implement functions that allow you to determine which bits of the pixel are allocated to each component. These are *int getAlphaMask()*, *int getRedMask()*, *int getGreenMask()*, and *int getBlueMask()*. The *RGBDefault* color model is a *DirectColorModel*.

In general, eight-bit samples correspond to palette-based color models and 32-bit samples correspond to direct color models such as 32-bit RGB. However, a *ColorModel* is a flexible representation that can be used to convert between pixel values and red, green, blue, and alpha components in any way that may be appropriate. For example, a topological map image could be stored as an array of *ints* with each sample representing the elevation of its location. A color model could then be created, which mapped the elevation to some standard topological map colors. This color model would be an indexed color model, however since the samples are stored as integers and not

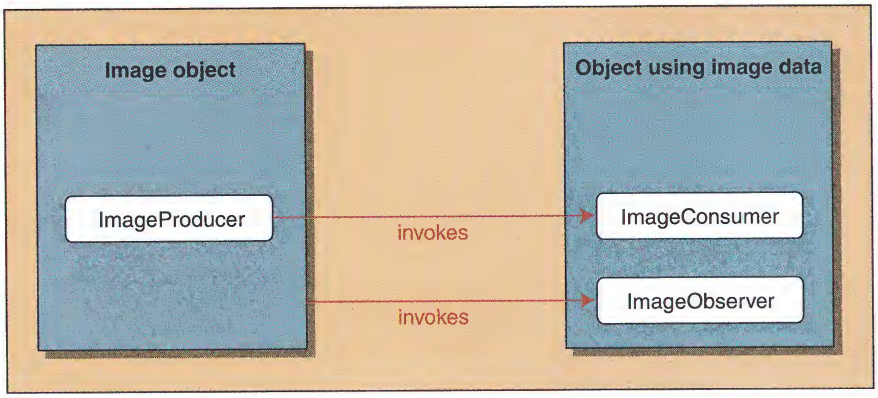


Figure 1: *Relationship between the basic classes and interfaces of the Java imaging model.*

(a)
```
void setPixels( int x, int y,
     int w, int h, ColorModel
     cm, byte[] pixels, int
     offset, int scansize);
```
(b)
```
void setPixels( int x, int y,
     int w, int h, ColorModel
     cm, int[] pixels, int
     offset, int scansize);
```

Example 2: *Function prototypes of the two* setPixels() *functions.*

bytes, it could have many more than 256 entries and thus very fine shading of color to represent changes in geography.

Image Processing

Image processing with Java is accomplished using image filters, which are objects derived from the *ImageFilter* base class. The *ImageFilter* class implements the *ImageConsumer* interface to receive the pixel samples for the image to be processed. An *ImageFilter* has a protected member variable, *consumer*, which refers to the *ImageConsumer* downstream of the image filter. Essentially, an *ImageFilter* is an *ImageConsumer* that manipulates the incoming pixel data and forwards the results to another *ImageConsumer* object.

As the image filter processes the samples received in its *setPixels()* function, the filter sends the processed samples downstream by passing the samples to the *setPixels()* function of its consumer. The image filter can pass the pixels to its consumer as it receives and processes the samples, or the filter can save the raw or processed pixels in a buffer and send them all at once to the consumer when the filter's *imageComplete()* method is called. The default implementation in the *ImageFilter* base class is a null filter. For each call to *setPixels()*, the base class simply calls the corresponding function in the consumer without modifying the samples. The rest of the methods inherited from the *ImageConsumer* method are similarly forwarded to the downstream consumer.

The Java API includes a few built-in filters to handle common tasks. The *ReplicateScaleFilter* will stretch or shrink an image to a given size by dropping or duplicating samples. The *AreaAveragingScale* filter resizes an image using a bilinear interpolation algorithm. The *CropImageFilter* extracts a rectangular subimage from the original image.

The *FilteredImageSource* class

Given that you have an instance of an *ImageFilter* object, how do you apply it to an image? This is where the *FilteredImageSource* class comes in. An instance of the *FilteredImageSource* class is an *ImageProducer* that creates a new image source from a given *ImageProducer* and an *ImageFilter*. The resulting image source produces an image which has been processed by the image filter.

Example 3 should make this clear. First, an image is loaded from the local file system using the default toolkit. Then an *AreaAveragingScaleFilter* is created which will scale images to half the size of the original loaded image. Next, a *FilteredImageSource* object is created from the scaling filter and the *ImageProducer* of the original image. Finally, an *Image* object is created from the new *FilteredImageSource*. Basically the *FilteredImageSource* constructor adds an image consumer/source pair in between the original *ImageProducer* and any future consumers. This process can be repeated any number of times to create an image filter chain or "pipeline;" see Figure 2.

It is important to realize that the image data is not actually filtered until it is requested by a consumer. Typically this is done implicitly when the image is displayed by calling *drawImage()*, although any *ImageConsumer* can start the pipeline flowing by calling *startProduction()* on the *ImageProducer* at the end of the pipeline. Notice that downstream *FilteredImageSource* objects forward delivery requests upstream. Eventually the request arrives at an image source that has image data ready to deliver and this *ImageProducer* object invokes methods on its registered consumers to push image data through the pipeline.

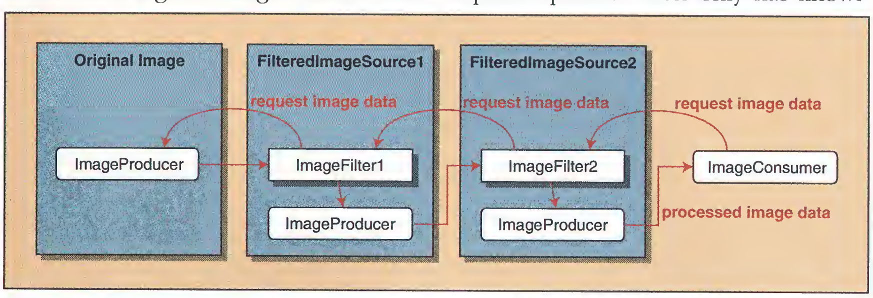


Figure 2: *Image processing pipeline.*

Image Filters

There are an infinite variety of filtering operations that can be applied to images in order to enhance or modify them. For the purposes of image processing with Java, filters can be classified according to the information required in order to process each sample. Here, we will classify image filters as point operations, geometric operations, or neighborhood filters.

Point Filters

The simplest filters process each sample independent of the surrounding samples. These filters can be simple color modifications, or can be dependent upon the pixel's location in the image. In any case, a point operation filter only has knowledge of a single pixel at a time. Contrast enhancement, color inversion, and dither filters, among others, can be written this way. This type of filter is so common that the Java API includes a special base class, *RGBImageFilter,* which makes implementing point operation filters very easy.

The *RGBImageFilter* class implements all the necessary *ImageFilter* machinery to filter an image pixel by pixel. One abstract function, *int filterRGB(int x, int y, int rgb),* needs to be implemented. This function is called for each pixel in the image and passed the pixel location as well as the 32-bit default RGB color model sample value. The *filterRGB()* function in the derived class should return the processed pixel value using the 32-bit default RGB color model.

Listing One (listings begin on page 124) is the implementation of the *GreyOutImageFilter*, which demonstrates the use of the *RGBImageFilter* class. This filter simply replaces half of the original image pixels with grey using a checkerboard pattern. This is an effect which you may want to use to make an icon or button appear inactive. Notice how short the implementation is. All the hard work has been done for us in the base class.

GreyOutImageFilter also demonstrates the proper way to handle the *canFilterIndexColorModel* protected member variable. For indexed color images, *RGBImageFilter* attempts to be more efficient by filtering only the color table, not the entire image. If the results of the filter are only dependent upon the color of each filter, the derived class should set this member to *true*. If the results depend upon the pixel location, as they do in the *GreyOutImageFilter*, then *canFilterIndexColorModel* must be set to *false*.

Geometric Filters

Another kind of filter moves pixels from one location in an image to another. These are known as geometric transformations. This is a very broad category and includes many types of filters including horizontal and vertical flipping, mirroring, and rotations. The Java API does not provide a specialized base class for implementing geometric operations like it does for point operations. We need to derive geometric transform filters from ImageFilter.

Listing Two implements a filter that rotates an image 90 degrees clockwise. An $m \times n$ pixel image is transformed into an $n \times m$ pixel image, with the pixel value occupying the original upper-left now occupying the upper-right corner.

The code is longer than that of the previous filter because we need to override several functions of the ImageFilter base class. To properly rotate the image, we need to save the width and height passed into

setDimensions(), and pass appropriate hints to the base class *setHints()*. Notice that after processing these values, you invoke the superclass implementation. For all of the functions in the *ImageFilter* class except *setPixels()*, you should invoke the superclass implementation to properly initialize the base class, passing either the original parameters or values modified appropriately for your implementation. Calling the superclass also ensures that the information is passed along to downstream consumers.

Image rotation is accomplished by taking each row of the rectangle of pixels received in *setPixels()* and passing it to the consumer as a column. We do some coordinate transformation to ensure that the pixels wind up in the correct location. Since the filter typically passes a column of pixels to the consumer, we clear the COMPLETESCANLINES and TOPDOWNLEFTRIGHT flags before we pass the hints along.

Neighborhood Filters

The most complex kind of filter that we will deal with here is the neighborhood filter. Each output pixel is a function of several nearby input pixels. Image sharpeners, smoothers, high-pass filters, and general convolution all fit into this category. Usually the pixel neighborhood is a small window of pixels centered on the pixel being processed. Common window sizes are 3×3 and 5×5. There are also "separable" image filters which process an image, a row, and then a column at a time. In this case, the neighborhood is composed of nearby pixels in the same row or column as the target pixel.

The implementor of a neighborhood filter must decide how to handle the image boundaries where there is not a full window of pixels to process. Common strategies are to leave the boundary pixels unprocessed, or to fill in the missing window samples by reflecting the image over the edge. The primary reason why implementing neighborhood filters with the *ImageFilter* class becomes complicated is that each call to *setPixels()* may pass only a small subrectangle of the image. The pixels at the edges of the subrectangles will not be delivered along with the neighborhood pixels required to process them. The easiest strategy to work around this problem is simply to save the image pixels in a private buffer as they are delivered to *setPixels* and process the image all at once when *imageComplete()* is called. The fully processed image can then be delivered with a single call to the consumer's *setPixels()* method.

The *SharpenImageFilter* (available electronically; see "Resource Center," page 3) uses this method to implement a 3×3 sharpening filter. Notice that the image to be processed is separated into color components and stored in four arrays named red, green, blue, and alpha. This allows the filter to sharpen both direct and indexed color model images.

```
Image big_image = getDefaultToolkit.getImage("lena.jpg");
ImageFilter shrinker=new AreaAveragingScaleFilter(big_image.getWidth()/2,
                                                  big_image.getHeight() / 2);
ImageProducer source=new FilteredImageSource(big_image.getSource(),
                                             shrinker);
Image small_image=createImage(source);
```

Example 3: *Using an* ImageFilter *with the* FilteredImageSource *class.*

Chaining Filters Together

Applying an *ImageFilter* to a given *Image* with the *FilteredImageSource* creates an image pipeline. If another filter is then applied to the resulting image, the new *FilteredImageSource* is added to the front of the pipeline. Each time the image is processed it is pushed all the way through the pipeline. If an image is filtered, and then displayed, applying another filter to the image results in repeating processing that has already been done as the original image source is sent all the way from the beginning of the pipe, through the previous filters and finally through the newest filter. Therefore when applying *n* filters in sequence to an image and displaying each intermediate result, the image is actually processed $n(n + 1)/2$ times.

The Java image processing API is flexible and written this way to handle changing source images. For example, if the original image source were video and not a still image, sending each frame though the entire pipeline would be the proper thing to do. However, it can be very inefficient when applying a series of filters to a still image in an interactive application.

To overcome this unnecessary overhead, the *BufferingFilteredImageSource* class (available electronically) can be used just like the *FilteredImageSource*. For still images, it buffers the result of applying its *ImageFilter* to the original image. When a downstream filter requests image data the *BufferingFilteredImageSource* delivers the samples from its internal buffer instead of forwarding the request upstream. For animated GIFs and other image sequences, it functions identically to the *FilteredImageSource*. Using this class makes applying a sequence of filters, one at a time, much more efficient since each filter is applied to an image only once.

To keep the implementation as simple as possible, the *BufferingFilteredImageSource* internally uses the *FilteredImageSource* class to actually perform the image filtering. An instance of *MemoryImageSource* is created and stored in the source member and functions as the *ImageProducer* for the buffered, filtered image. If the image is animated, then no buffering is done and the *FilteredImageSource* instance is used as the source instead. *BufferingFilteredImageSource* implements the *ImageProducer* interface simply by forwarding the calls to the source member object.

The only complexity in the implementation is the *BfisInternalConsumer* class that the *BufferingFilteredImageSource* uses to save the processed image in a buffer. The buffer is then used to create the *MemoryImageSource*. Since the *FilteredImageSource* object may deliver the filtered image synchronously or asynchronously, we need to allow for both cases. This is done by synchronizing the *BufferingFilteredImageSource* constructor with the *imageComplete()* function of the *BfisInternalConsumer* using the object and the *done* flag.

The JIPTestApplet

To demonstrate the filters and classes presented in this article, use the test applet, JIPTestApplet (also available electronically), along with JIPTestApplet.html, which is an HTML page used to demonstrate the applet. The applet can be configured to display and process up to ten images. The filters given in this article can be applied repeatedly using either the *FilteredImageSource* or *BufferingFilteredImageSource*. Applying several filters in succession to a still image will demonstrate the efficiency gained by using the *BufferingFilteredImageSource*. Using the *FilteredImageSource*, successive filter applications take longer and longer. Using the *BufferingFilteredImageSource*, each invocation takes the same amount of time. You can also use the test applet to explore the results of applying several filters to an image, for example, the *GreyOutFilter* followed by the *SharpenImageFilter*.

Conclusion

The Java image-processing model is powerful, flexible, and expandable, and can be used to create complex image-processing applications. Image processing is a computationally intensive activity. With a JIT compiler, Java performance is more than adequate for basic processing of moderate sized images. For interpreted Java virtual machines or complex multipass computations, the processing should be kept to small images.

DDJ

(Listings begin on page 124.)

BOSTON Premiere!

See You At The Summit

Web Design & Development '98

6 focused tracks examine the trends and give you the latest tools and techniques:

- Strategy
- Information Design
- Visual Design
- Usability
- Programming
- Backend

Being a Web Professional Requires Some Remarkable Skills

It takes your entire team working together. To stay on top, you need solid training from the best minds in the business. From the fundamentals of site usability to cutting-edge programming techniques, Web Design & Development '98 sets the standard for giving web professionals what they need to succeed. Web '98 provides practical training from the leading experts in information design; dynamic HTML vs. Java; XML; privacy; metadata; browser hell and more. Visionary keynotes. Peer networking and brainstorming. And an exhibition of powerful tools from the hottest companies around. Join us for the most intense gathering of web brilliance the East Coast has even seen. Because when you're serious about your job, there's only one place to go.

www.mfweb.com

Feeling nostalgic? Contact us by phone at 800-441-8826 or by e-mail at web98@mfi.com.

HYNES CONVENTION CENTER
BOSTON, MA

September 22-26 1998
CONFERENCE

September 23-25 1998
EXHIBITION

WEB98
design&development
where web professionals converge

Image Segmentation for Image Recognition

Lee Kamentsky

Good ideas reappear in surprising places. When Lee first told me about his algorithm, I thought he was discussing an old flood-fill algorithm—one that I remembered seeing long before it appeared in Kent Porter's "Graphics Programming" column (*DDJ*, June 1989). There are good reasons for the similarity: Both algorithms rely on a fundamental insight that applies to many data-analysis tasks—where possible, work with groups of data, not with single elements.

Rather than studying a single pixel at a time, Lee's technique immediately bundles pixels into larger structures. As in many graphics problems, the natural unit is a horizontal line of pixels. By quickly moving the problem from one involving pixels to one involving lines, Lee reduces his work by an order of magnitude.

—Tim Kientzle

Subtle variations of color and texture in photographic images make it difficult to clearly identify a foreground and a background. It's considerably simpler with "artificial" images, such as scanned text. The relatively high contrast makes it easier to separate pixels, but you still have to somehow identify regions of pixels and extract relevant information. In this article, I'll describe an algorithm that quickly reduces high-contrast images to a data set that lends itself to image recognition and analysis. This algorithm could be used to analyze black text on a white background, calibration marks on a surface, red Legos in a toybox, or a slide containing blots of a drug being tested by a pharmaceutical company.

The initial stage of image analysis usually consists of two steps—thresholding and segmentation. Thresholding algorithms classify pixels as either foreground or background, or classify pixels according to their foreground class (green, red, and blue classes, for instance, or skin texture, hair texture, and clothing texture). Thresholding algorithms can extend past analyzing the raw brightness or color of a pixel. Some involve linear transforms such as convolution filters and FFTs. Others rely on nonlinear transforms such as image dilation.

Segmentation algorithms group similar pixels together into coherent units. Segmentation is often regarded as a backwater of image analysis. However, a good segmentation algorithm can yield easily analyzable, abstract objects with a minimum of processing. The algorithm I describe here has good performance. It visits each pixel exactly once, processing the pixels in their raster order, which maintains cache coherency, improves bus bandwidth and eliminates disk thrashing. It yields data structures representing connected groups of pixels; the data structures can be manipulated to extract information without revisiting each pixel.

The algorithm groups pixels into run-length encoded lines. It then groups the lines into structures I call "blobs." The blobs contain groups of lines that overlap. This yields an object that describes a set of four-connected (north, south, east, and west) pixels, none of which are connected to any pixel outside of the set. You can quickly find a blob's extent, position, center, or area. You can analyze the blob's topology and shape. You can revisit the pixels within the blob to determine their brightness, coloration, or texture. These techniques lend themselves to the tasks of optical-character recognition (OCR), scene recognition (where the scene consists of artificial, high-contrast objects), and automated chemical analysis.

Lee is a software engineer at CompuCyte Corporation. He can be contacted at leek@msn.com.

The Segmentation Algorithm

Suppose you're trying to analyze black text against a white background. In this context, the segmentation algorithm looks at one raster at a time and breaks it down into a series of black lines. Each line is compared to the lines on the previous raster (the "old line list") and is either added to an existing blob or a new blob is created.

If a blob has lines on the old line list, those lines are open lines. I maintain a count of open lines for each blob. When the count falls to zero, that blob is finished and need no longer be considered. The bulk of the algorithm, then, consists of walking down the old line list and comparing new lines to see how to attach them to blobs.

The algorithm starts with the genesis of a new line. The application typically maintains a threshold and scans until it finds a pixel whose value is over the threshold. It then starts a line and scans until it hits a pixel under threshold. The application then builds a line consisting of the X start, X end, and the current Y. The line also contains a pointer to its blob (null signals that the line is unattached) and a pointer to the next line in the blob.

There are three possible relationships between the current old line and new line, as illustrated in Figure 1.

- The old line might end before the new line begins; see Figure 1(a).
- The new line might end before the old line begins; see Figure 1(b).
- The two lines might overlap; see Figure 1(c).

If the old line is wholly before the new line, it is finished. I decrement its blob's open line count and, if the open line count has reached zero, add the blob to the list of finished blobs. I then advance to the next old line in the list. If the new line is wholly before the old line, then we have finished attaching the new line. If no previous old line overlaps this new line, then the new line starts a new blob.

The third case involves overlapping old and new lines. Its implementation requires two parts: attaching and ending. You attach the new line to an old line in one of three ways. If the new line is not yet linked to a blob, we link it to the old line's blob. If the new line is already linked to the old line's blob, then you increment the blob's loop count (this is a signal that the blob split into two halves and these two halves have been rejoined by the new line, like at the bottom of the letter "O"). If the new line has already been linked to a blob other than the old line's blob, then the new line joins two blobs (like at the bottom of the letter "V"). You merge the two blob's line lists and discard one of the blobs.

Merging blobs is an important feature of any segmentation algorithm. Figure 2 illustrates a particularly difficult example, in which three new lines combine four separate blobs. You need to compare the ends of the new and old lines after attaching the new line. If the old line ends first, you can move onto the next old line. If the new line ends first, then you have to increment the blob's open line count (lines can now be attached to both the old and the new line). You then move on to create the next new line.

The algorithm has one additional minor complexity: At the end of a raster, you either run out of new lines or old lines. If you run out of new lines, you run through a loop that ends the remaining old lines. You decrement the open line counts and finish any blobs whose open line count falls to zero. If you run out of old lines first, you must create new blobs for each new line until the end of the raster.

Additional Features

It's possible to augment this algorithm to collect additional information about each blob. I've already included the loop counts, since those are so easy to extract. The demonstration program makes use of the loop count to distinguish the loops in bullseyes: The bullseye rings must be closed to be recognized. You might also use the loop count to help in distinguishing letters; uppercase "B" and lowercase "g" are the only letters with two loops (assuming no malformations).

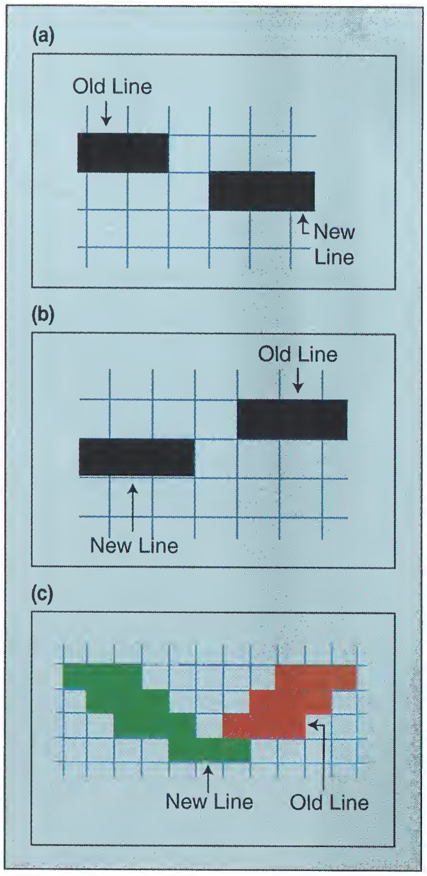


***Figure 1:** Possible relationships between old and new lines: (a) Old line ends first, advance to next old line; (b) new line ends first, advance to next new line; (c) old and new lines overlap, add new line to blob.*

Another group of useful features are the image tops and bottoms. The tops are those lines that were not attached to any line above. The total number of tops is one more than the number of merges required to produce the blob. The bottoms are those lines that had no other line attached below them. The algorithm can find these by setting a flag when first considering an old line. The algorithm clears the flag if a new line attaches to the old line. You check the flag when finishing the old line; if it is still set, you have a bottom.

Numbers of tops and bottoms by themselves can be used to distinguish shapes. For instance, the letter "w" has three tops and two bottoms for both serif and sans-serif fonts, the small "m" has three tops and three bottoms. Positions of tops and bottoms are perhaps more telling; a top directly above a bottom may indicate a vertical line (in character recognition, this is typically the leftmost stroke of the letter). It's quite possible to build a neural network character recognizer that uses a number of loops, top position, and bottom position as its input features; this recognition engine is efficient and provides a confidence output along with its judgment.

Positional information can be recorded during segmentation. The algorithm can update the enclosing rectangle for the blob as it adds lines (the top of the blob is always one of its starting lines, the bottom is always the old line at which the line count drops to zero). The algorithm can also compute the blob centroid (if each pixel weighed the same, the centroid would be the blob's center of mass) and area. It may be more efficient, however, to compute these during a quick trip through the blob's line list after it has been finished.

Optimizations

Ironically, the two most expensive parts of the segmentation algorithm are outside of the algorithm proper. Thresholding is the most expensive, because it requires examining each individual pixel. Using table lookups or other techniques that examine several pixels at a time can help to speed this part. A quick optimization may be to check for all ones or all zeros. Typically, though, thresholding is sped up through hardware assist: The data is thresholded, run-length encoded, and transmitted in encoded form.

The second most expensive operation is allocation. Artifacts of image capture can often cause a jagged edge at the top of an object, The algorithm may create a blob initially for each of these jagged lines. Blob creation can be an expensive operation, even if the blobs are unlinked from a private heap. It's possible to modify the algorithm so that blobs are created only when a previously unattached new line is attached to a previously unattached old line. This delays blob creation past typical effects caused by aliasing. A single new line joining many unattached old lines will only create a single blob.

My implementation maintains an array of pointers to lines. This indirection can be eliminated by allocating all lines from a single array. You can then set an array index to the first line on the old line list and fetch subsequent old lines by incrementing the index. It's possible in many operating systems to reserve a large block of memory without actually allocating the backing store necessary to maintain the virtual memory image on disk (this is the *VirtualAlloc* function in Windows 95/NT). The algorithm can reserve the theoretical maximum size memory required for the worst case (which is half the number of pixels in the image times the size of a line). The algorithm can perform the actual allocation at the beginning of the raster, enlarging the array of lines to prepare for the worst case. This eliminates the need to check for "out of memory" when allocating lines during the course of a scan.

Finally, many processors have a sweet spot at 16 bytes when dealing with arrays of structures. (Typically, the bus is 32-bits wide and the bus bursts are four words long. A 16-byte structure makes optimum use of one burst.) The line structure consists of an X start, X end, Y coordinate, pointer to the next line in the blob, and pointer to the line's blob. That's 20 bytes if the algorithm uses 32-bit integers and pointers. Packing the X start and X end into two 16-bit words may provide a significant boost in speed.

Applications

The structures yielded by the segmentation algorithm lend themselves to a variety of uses. The lines provide compression for a number of tasks. First among these are tasks involving the relations of the blobs to each other and to the background. It's an easy matter to relate the blobs to their neighbors once they have been segmented; you can form agglomerations (such as

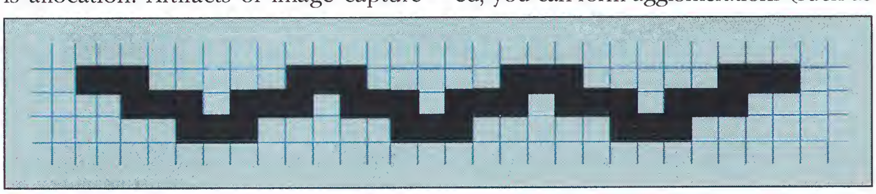


Figure 2: *An example of the difficult "w" problem.*

lines of print or areas of half-tone dot images) that provide useful feature information. My most recent use of the algorithm has been a bullseye recognizer that detects a microscopic bullseye deposited at an exact spot on a microscope slide by photolithography. The segmentation algorithm quickly identifies the two rings and target of the bullseye by searching for the only three objects that have centroids that are close together (the middle of each ring corresponds with the middle of the bullseye).

It's possible to perform template matches using the algorithm data structures. (I've included a template matcher in Listing One; listing begins on page 124.) A template matcher can match blob against blob. This is done by relying on the line order within the blobs to efficiently find which parts of the image overlap the template. The resulting score gives a confidence measure that the template is the same shape as the image. Template matching can be used in OCR to discriminate between letters that have similar overall shapes (such as "A" and "R," which both have one loop, one top, and two bottoms).

The segmentation algorithm can be used to find and outline objects deposited on a substrate. For instance, a drug manufacturer might test thousands of variants of a drug at once using a robotic deposition tool that places the results of different reactions at different spots on a dish. The spots can be found using the segmentation algorithm and the chemical reaction can then be analyzed by measuring the intensity and color of the reactions in the spot. The algorithm can be used in other aspects of robotics; a few marks on an object can be used to track the object in real time and position it. Again, these algorithms work best in an artificial environment in which lighting and contrast are controlled to maximize the difference between foreground and background.

The three parts of the bullseye have common centroids

Conclusion

This segmentation algorithm rapidly identifies the boundaries of contiguous objects from their background. It makes efficient use of modern CPU resources and yields objects with data structures that have utility. I believe that it could serve as the core of a number of interesting image processing applications. I would like to see it used in novel ways, such as in identifying— in real time— high contrast objects placed in video scenes as markers for animated characters. It might also be used as the base segmentation algorithm for real-world imaging using sophisticated image processing techniques that enhance foreground. The algorithm can be extended to three dimensions by matching against two old line lists: one in the X–Y plane and one in the X–Z plane. This adaptation might be used in computer-aided tomagrophy to identify tumors. In any case, the algorithm is dependable and useful and should serve you well in appropriate applications.

DDJ

(Listing begins on page 124.)

SD '98 COM BOOTCAMP

Seminars in Building COM/DCOM Components with Visual C++, MFC and ATL.

5 Days of Intensive Hands-On Training

Chicago	Aug 31—Sept 4
Seattle	Oct 5—9
San Jose	Oct 12—16
Arlington, VA	Nov 9—13
Boston	Nov 16—20
Atlanta	Nov 30—Dec 4

Enlist in a five day, hands-on SD '98 Bootcamp and learn how to build powerful and flexible programs with COM and DCOM.

WHAT YOU'LL LEARN

- The COM Philosophy: what it is, how it works, and why you should implement all of your significant software projects using COM.
- How to implement COM Objects in C++ using MFC and ATL.
- How to implement helper objects and use custom data types without custom marshaling.
- How to build ActiveX controls with MFC and ATL.
- How to build COM clients in C++ by importing server type libraries.
- How to add callback support, connection points, aggregate objects, tear-off interfaces, and hierarchies of objects using MFC and ATL.
- How to test and deploy COM objects remotely using DCOM.

REASONS TO ATTEND

- You love COM.
- You want to love COM.
- You want to write COM Objects in MFC and ATL.
- You hate monolithic projects.
- You know that COM will be THE influence in software development for the next 5 years and you want to be part of it!

EXPERT INSTRUCTOR:

Richard Hale Shaw

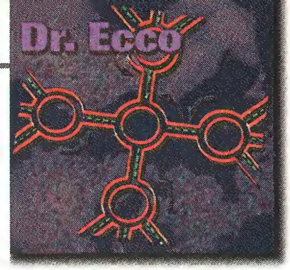
Shaw is a world-renowned authority on COM and Windows programming with C++. He's written for *PC Magazine*, *Microsoft Systems Journal*, and *VC++ Developer's Journal*. He's also the COM Track Chair for the Software Development Conference. (www.RichardHaleShaw.com).

TEAM DISCOUNT!
Register 5 more people at the full price and the 6th person attends FREE! All registrants also receive a free SD collectors t-shirts!

SD|98 software DEVELOPMENT

Miller Freeman

Contact us today! + www.sd98.com/bootcamp + 800-441-8826 + 415-905-2702 + bootcamp@mfi.com



Lines of Fire

Dennis E. Shasha

He claimed to be a military man, but Captain Solo didn't seem to have the right bearing. He slumped in one chair with his feet propped on another. His uniform was clean, but badly wrinkled. His hair was unkempt. In brief, he looked like one of my colleagues.

His smartly dressed assistant, Lieutenant Hood, presented the problem to Ecco. "Here is the layout of the valley, sir," he said.

"Our hovertanks will be approaching from the eastern pass. When our adversaries detect us, they will approach from the western pass.

"Our job is to destroy the underground factories in the valley. The factories themselves have no defenses except their one-meter-thick walls, but we must quickly dominate the ground above. That means we must rapidly reach a situation in which we can destroy any new vehicle that comes into the valley."

"Why can't you just cross over with your first hovertank and block their pass?" Liane asked.

"They have field guns and rocket-propelled explosives inside bunkers overlooking the western pass," the lieutenant responded. "We have, however, located 25 hills that are good vantage points for our hovertanks. Occupying any 10 of them will allow us to destroy the underground factories. We just want to be safe on them while keeping our adversaries to under five."

"So, drive to the first 10 hills you can see," Ecco said with some impatience.

Solo spoke up. "We wouldn't be here if it were that simple, Dr. Ecco. You must think of this as a two-person game with alternating moves. We place a vehicle, then they place one. Then we place one. Then they do. And so on. We just want to be sure that after they have placed five, they can't place any more, whereas we can place at least 10 altogether."

"Can you get to any hill equally fast?" Liane asked.

Solo nodded. "I designed every hovertank to fly at 80 miles an hour and to stop on a dime."

"Unfortunately, our adversaries have access to the same technology," added the lieutenant.

Dennis, a professor of computer science at New York University and author of The Puzzling Adventures of Dr. Ecco *(Dover, 1998), can be contacted at DrEcco@ddj.com.*

"Our biggest advantage is that we will be in the valley first. In fact, we hope that we will be able to place at least two and possibly three hovertanks on hills in the valley before they place any of theirs. After that, we will alternate. Remember, we want to prevent them from placing more than five of their hovertanks and we want to be able to place at least 10 of ours. Also, our 10 hovertanks must be able to fire upon all unoccupied hills. Finally, their five hovertanks must not be able to fire upon our 10."

"Can you tell me which hills can fire upon which others?" Ecco asked.

"Yes," the lieutenant responded, handing Ecco the table below (also available electronically, see "Resource Center," page 3). "We've numbered the hills from 0 to 24. I hope the table is clear. For example, hill 0 can fire upon hills 1 2 3 7 9 8 13 17 18 19 20 22 24. Note that firing is usually but not always symmetric. For example, 0 can fire upon 2 but the reverse does not hold."

Fire From	Fire Upon
0	1 2 3 7 9 8 13 17 18 19 20 22 24
1	0 2 3 9 10 13 14 15 16 17 20 21 24
2	3 5 9 11 12 13 16 22 23
3	2 5 7 9 11 12 13 16 22
4	5 7 9 11 12 13 16 22 23
5	2 3 4 7 9 11 12 23
6	1 2 3 7 9 8 13 14 15 16 18 20 21
7	3 4 5 9 12 19 20 22 23
8	0 6 10 14 15 17 18 19 24
9	2 3 4 5 7 11 19 20 22
10	1 2 3 7 8 9 13 15 16 20 21 22 24
11	2 3 4 5 9 19 20 22 23
12	2 3 4 5 7 8 17 18 23
13	0 1 2 3 6 7 9 10 16 18 21 22 24
14	1 6 8 15 16 17 18 19 21 22 24
15	1 2 3 6 8 10 14 16 18 19 20 22
16	1 2 3 4 7 6 9 10 13 14 15 17 19 22
17	0 1 8 14 16 18 20 21 24
18	0 6 8 13 14 15 17 21 22
19	0 8 14 15 16 20 21 22 24
20	0 1 2 3 6 9 10 15 17 19 22 24
21	1 6 10 13 14 17 18 19 24
22	0 10 13 14 15 16 18 19 20
23	2 4 5 7 11 12 19 20 22
24	0 1 8 10 13 14 15 17 19 20 21

Ecco and Liane studied the table for a while. Suddenly, Liane giggled and said, "Two sets of queens problems." She then sketched something on paper that looked like two squares and showed it to Ecco who nodded.

Ecco then turned to us and stifled a yawn. "Gentlemen, let me take a three, four, or five minute nap, and then I'll get back to you."

When Ecco returned, he handed Captain Solo a piece of paper. "If you can place three hovertanks before your adversaries can place any, then take these three and you can prevent the adversary from occupying any hills, because your hovertanks will be able to fire upon every remaining hill. If you can place only two before they can place any, then choose these two. You will be able to keep the adversaries to under five, occupy 10 yourself, and be able to fire upon all remaining hills. If you can place only one before they can place any, then I don't think you can achieve your goals, but I'm not sure and, well, you haven't asked."

Reader: Please show Solo how (a) he can dominate the valley entirely by occupying three hills before his adversaries can occupy any; and (b) he can achieve the conditions of the problem by occupying two hills before his adversaries can occupy any (and alternating thereafter). If you have a proof either way concerning the last problem (if the adversaries allow Solo only one hill before alternating moves begins), then send it to me at DrEcco@ddj.com.

Last Month's Solution

Since each accusation must include at least one lying informant and all two accusation pairs below are disjoint, there have to be at least six:

Accuser	Accused
petra	gwenyth
sam	larry
dave	mike
isaac	nick
hillary	kris
olivia	ulm

If the following eight people have been turned, they suffice to explain all the accusations that commissioner Bratt has presented to Ecco: isaac hillary larry olivia gwenyth dave sam petra. Ecco doesn't know whether this is the minimum possible.

Superheurists: *DDJ* readers Gary Knowles, Jon Beal, and Kent Donaldson all found clever solutions to the May 1998 Nimmerics problem. Kent also has the best solution so far to the Territory Game in which his first six ships get 66.60 percent of the valley and all seven get 73.83 percent.

DDJ

JAVA IMAGE PROCESSING

Listing One

```
import java.awt.Color;
import java.awt.image.*;

public class GreyOutImageFilter extends RGBImageFilter {
    // save the int value of the grey color...
    protected int greyOutValue = Color.gray.getRGB();
    public GreyOutImageFilter() {
        // this filter is position dependent, so we can't filter
        // by just changing the color table...
        canFilterIndexColorModel = false;
    }
    public int filterRGB( int x, int y, int rgb) {
        // set every other pixel to grey using a checkerboard pattern...
        if (((x ^ y) & 1) == 0) {
            return greyOutValue;
        }
        else {
            return rgb;
        }
    }
}
```

Listing Two

```
import java.awt.image.*;

public class RotateClockwiseImageFilter extends ImageFilter {
    protected int srcwidth, srcheight;
    protected int destwidth, destheight;
    public void setDimensions( int width, int height) {
        // source height becomes destination width and vice-versa...
        this.destwidth = height;
        this.destheight = width;

        // tell the consumer the size of the image that we will be sending...
        super.setDimensions( this.destwidth, this.destheight);
    }
    public void setHints( int hints) {
        // because this filter delivers pixels a scan column at a time,
        // we need to clear COMPLETESCANLINES and TOPDOWNLEFTRIGHT hint
        // bits and set the RANDOMPIXELORDER bit...
        hints = (hints & (~COMPLETESCANLINES) &
                              (~TOPDOWNLEFTRIGHT)) | RANDOMPIXELORDER;
        super.setHints(hints);
    }
    public void setPixels( int x, int y, int w, int h, ColorModel cm,
                              int[] pixels, int offset, int scansize) {
        // start is the offset into buffer of first pixel of current source
        // row, which will become the top pixel in destination column...
        int start = offset;

        // destx is zero-indexed destination column of source pixel row...
        int destx = destwidth - y - 1;

        // send the pixels on to the consumer, each row of the source image
        // becomes a 1-pixel wide column of pixels in destination image...
        for (int j = 0; j < h; j++) {
            consumer.setPixels( destx, x, 1, w, cm, pixels, start, 1);
            start += scansize;
            destx--;
        }
    }
    public void setPixels( int x, int y, int w, int h, ColorModel cm,
                              byte[] pixels, int offset, int scansize) {
        // start is the offset into the buffer of the first pixel of current
        // source row, which will become top pixel in destination column...
        int start = offset;

        // destx is zero-indexed destination column of source pixel row...
        int destx = destwidth - y - 1;

        // send the pixels on to the consumer, each row of the source image
        // becomes a 1-pixel wide column of pixels in destination image...
        for (int j = 0; j < h; j++) {
            consumer.setPixels( destx, x, 1, w, cm, pixels, start, 1);
            start += scansize;
            destx--;
        }
    }
}
```

ALGORITHM ALLEY

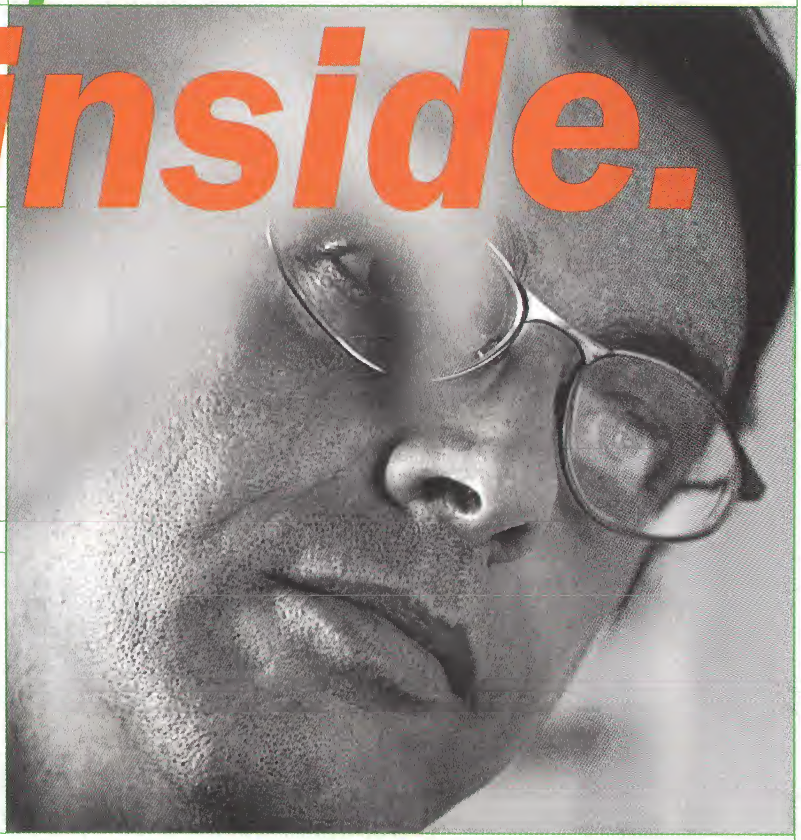
Listing One

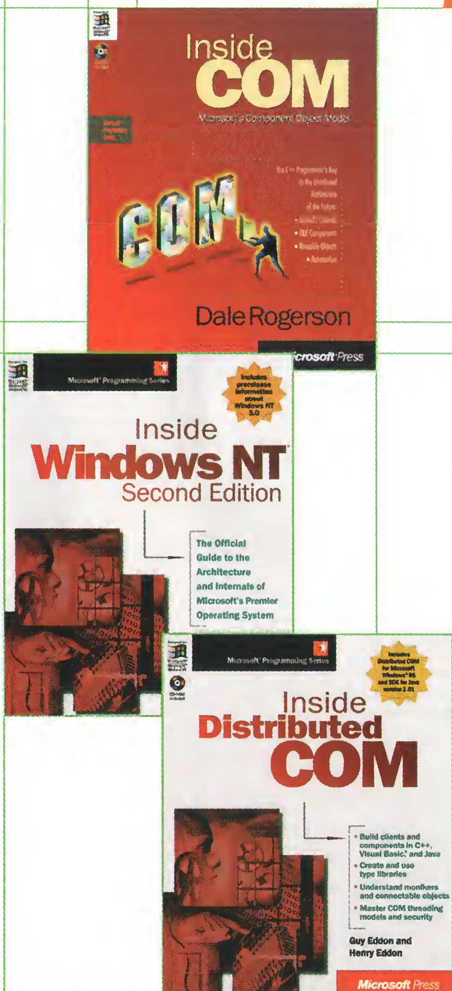
```
void CMonochromeBitmap::BuildBlobList()
{
  CLineVectorIterator clviNew,clviEnd;
  AllocateLineBlock(clviNew,clviEnd);
  m_pBlob.reserve(eBlobsPerBlock);

  // Create two vectors of pointers to lines to hold the new line list
  // and the old line list. We initialize them to hold Width()/2 members.
  // This is the maximum possible number of lines.
  CLinePointerVector clpvRaster[2];
  clpvRaster[0].reserve(Width()/2);
  clpvRaster[1].reserve(Width()/2);
  int nOldLineIndex = 0;
  int nNewLineIndex = 1;
  // At the start, the old line list has no members.
  CLinePointerVectorIterator clpviOldEnd = clpvRaster[nOldLineIndex].begin();
  for (int nY = 0; nY < Height(); nY++) {
    CLinePointerVector &clpvOld = clpvRaster[nOldLineIndex];
```

```
    CLinePointerVector &clpvNew = clpvRaster[nNewLineIndex];
    nOldLineIndex = nOldLineIndex ^ 1;
    nNewLineIndex = nNewLineIndex ^ 1;
    CLinePointerVectorIterator clpviOld = clpvOld.begin();
    CLine *pLineOld;
    if (clpviOld == clpviOldEnd)
      pLineOld = NULL;
    else
      pLineOld = *clpviOld++;

    CLinePointerVectorIterator clpviNew = clpvNew.begin();
    // Get the raster to process and set up for the first byte
    const BYTE *pRaster = GetRaster(nY);
    BYTE bBit = 0x80;
    BYTE bByte = *pRaster++;
    //  Set up the X extents.
    int nX = 0;
    const int nXEnd = Width();

    while(nX < nXEnd) {
      CBlob *pBlobOld;
      // Search for the start of a line.
      while((bByte & bBit) == 0) {
        if (++nX == nXEnd) goto linedone;
        bBit = bBit >> 1;
        if (bBit == 0) {
          bBit = 0x80;
          bByte = *pRaster++;
        }
      }
      // Start a line.
      if (clviNew == clviEnd) {
        // need more memory for lines.
        AllocateLineBlock(clviNew,clviEnd);
      }
      CLine &lineNew = *clviNew++;
      lineNew.m_nXStart = nX;
      lineNew.m_pNextLineSameBlob = 0;
      lineNew.m_nY = nY;

      // Put it on the new list.
      *clpviNew++ = &lineNew;
      // Find the extent of the line.
      do {
        if (nX == nXEnd) break;
        bBit = bBit >> 1;
        if (bBit == 0) {
          bBit = 0x80;
          bByte = *pRaster++;
        }
        nX++;
      } while (bBit & bByte);
      lineNew.m_nXEnd = nX - 1;
      // Now finish all old lines wholly before our new line
      if (pLineOld) {
        while (pLineOld->m_nXEnd < lineNew.m_nXStart) {
          pBlobOld = pLineOld->m_pBlob;
          if (--(pBlobOld->m_nOpenLines) == 0) {
            AddBlob(pBlobOld);
          }
          if (clpviOld == clpviOldEnd) {
            pLineOld = NULL;
            break;
          } else {
            pLineOld = *clpviOld++;
          }
        }
      }
      // Do the first line that overlaps our new line
      if (pLineOld && pLineOld->m_nXStart <= lineNew.m_nXEnd) {
        pBlobOld = pLineOld->m_pBlob;
        lineNew.m_pBlob = pBlobOld;
        *(pBlobOld->m_ppLastLine) = &lineNew;
        pBlobOld->m_ppLastLine = &(lineNew.m_pNextLineSameBlob);
        // See if the new line ends the old line or vice-versa.
        if (pLineOld->m_nXEnd > lineNew.m_nXEnd) {
          // the old line extends past the new line.
          // the new line gives the old line's blob another open line.
          pBlobOld->m_nOpenLines++;
          continue;
        } else {
          // we close the old line and extend the blob at the
          // same time. We continue to close old lines.
          if (clpviOld == clpviOldEnd) {
            pLineOld = NULL;
          } else {
            pLineOld = *clpviOld++;
            while(pLineOld->m_nXEnd <= lineNew.m_nXEnd) {
              // End this line too. We have two cases:
              CBlob *pBlobOther;
              if ((pBlobOther = pLineOld->m_pBlob) == pBlobOld) {
                // the old line is part of the new line's blob.
                // This is just a loop.
                pBlobOld->m_nLoops++;
                pBlobOld->m_nOpenLines--;
              } else {
                // This is the merge case
                pBlobOld->Merge(pBlobOther);
                delete pBlobOther;
                pBlobOld->m_nOpenLines--;
              }
              if (clpviOld == clpviOldEnd) {
                pLineOld = NULL;
                break;
              } else {
                pLineOld = *clpviOld++;
              }
            }
            if (pLineOld && pLineOld->m_nXStart <= lineNew.m_nXEnd) {
              // This last old line overlaps and ends the new line.
              CBlob *pBlobOther = pLineOld->m_pBlob;
              if (pBlobOther == pBlobOld) {
                pBlobOld->m_nLoops++;
              } else {
                pBlobOld->Merge(pBlobOther);
                delete pBlobOther;
              }
            }
          }
        }
      } else {
        // This is the case where no line overlaps the new line.
        // Start a blob with this line.
        CBlob *pBlobNew = lineNew.m_pBlob = new CBlob;
        pBlobNew->m_nOpenLines = 1;
        pBlobNew->m_pFirstLine = &lineNew;
        pBlobNew->m_ppLastLine = &(lineNew.m_pNextLineSameBlob);
        pBlobNew->m_nLoops = 0;
      }
    }
linedone:
    if (pLineOld) {
      while (TRUE) {
        // Finish all old lines.
        CBlob *pBlobOld = pLineOld->m_pBlob;
        if (--(pBlobOld->m_nOpenLines) == 0) {
          AddBlob(pBlobOld);
        }
        if (clpviOld == clpviOldEnd) break;
        pLineOld = *clpviOld++;
      }
    }
    // Finally, record the end of the new list as the
    // end of the old list to be.
    clpviOldEnd = clpviNew;
  }
  // At the end, we've got one last raster of lines to finish.
  CLinePointerVector &clpvLast = clpvRaster[nOldLineIndex];
  for (CLinePointerVectorIterator clpvi = clpvLast.begin();
  clpvi != clpviOldEnd;) {
    // Finish all old lines.
    CBlob *pBlobOld = (*clpvi++)->m_pBlob;
    if (--(pBlobOld->m_nOpenLines) == 0) {
      AddBlob(pBlobOld);
    }
  }
  // and finally, segmentation is complete
}
```

DDJ

The world's most perfectly developed software starts inside.





You supply the creative vision, and Microsoft Press helps you turn vision into reality with the definitive programming guides for Microsoft® Windows NT® and component development. *Inside Windows NT, Second Edition,* is the authoritative blueprint you need to unlock the full power and performance of the Windows NT platform. *Inside COM* is your clear and simple, practical guide to building elegant COM components. *Inside Distributed COM* helps you use Microsoft's Distributed Component Object Model to create powerful new business solutions based on multi-tier or even truly distributed applications. Be the ultimate insider and see what develops. Microsoft Press will help you

start faster and go farther.

Available in quality bookstores and computer stores worldwide. To locate your nearest source for Microsoft Press® products, reach us at: 1-800-MSPRESS in the U.S., or **mspress.microsoft.com**

Where do you want to go today?®

Microsoft® Press

mspress.microsoft.com

Displacement and Other Stories

Gregory V. Wilson

Software Visualization: Programming as a Multimedia Experience
John Stasko, John Domingue, Marc H. Brown, Blaine A. Price (editors)
MIT Press, 1998
562 pp., $55.00
ISBN 0-262-19395-7

C/C++ Software Quality Tools
Mark L. Murphy
Prentice Hall PTR, 1996
352 pp., $45.00
ISBN 0-13-445123-6

Perl: The Programmer's Companion
Nigel Chapman
John Wiley & Sons, 1997
292 pp., $34.99
ISBN 0-471-97563-X

Effective Perl Programming
Joseph N. Hall with Randal L. Schwartz
Addison Wesley Longman, 1998
288 pp., $32.95
ISBN 0-201-41975-0

Perl 5 Interactive Course
Jon Orwant
Waite Group Press, 1998
992 pp., $49.99
ISBN 1-57169-113-8

Software Runaways: Monumental Software Disasters
Robert L. Glass
Prentice Hall PTR, 1998,
288 pp., $28.00
ISBN 0-13-673443-X

Displacement is a wonderful thing. I'm two months away from the next major release of the product I'm responsible for, I have a training course on software engineering for scientists and engineers to write, and what am I doing? I'm watching Working Model (http://www.workingmodel.com/) simulate a stepping gear called a "Geneva wheel." Working Model is a GUI version of first-year physics: Arbitrary shapes can be sketched, connected, given physical properties such as mass and elasticity, and then set in motion to interact according to $F=ma$ and, um, gosh, it's been a long time since I did physics, hasn't it?

Once I've finished playing with gears, I'm going back to LCSI's MicroWorlds (http://www.lcsi.ca/), a multimedia version of the Logo educational programming environment. Logo is interactive, forgiving, and graphical, so that children can easily create and explore. In a better universe, it would have revolutionized the way computers are used in schools. This latest incarnation adds sound, web publishing, and a multitude of other new features, without obscuring its central simplicity. If you want your children to learn how to think with their computers, instead of just treating it as TV with hyperlinks, then MicroWorlds is definitely worth a look.

A couple of hours with either of these programs is enough to make most programmers ask themselves, "Why aren't my tools this easy to use?" According to the contributors to *Software Visualization*, edited by John Stasko, John Domingue, Marc H. Brown, and Blaine A. Price, part of the answer is that we make far too little use of interactive 2D and 3D graphics. Even the most advanced integrated-development environments (IDEs) are built around a text editor. It might highlight syntax and it might be accompanied by some list boxes or trees showing files and classes, but it's still just vi on steroids. If you compare this to the dozens of sketches that programmers draw to show data structures, inheritance, and control flow, you can see how much more we could be doing.

Computer-aided software engineering (CASE) tools have tried to integrate structured diagrams into IDEs. (A good entry-level example is Stingray's Visual CASE, at http://www.stingsoft.com/vcase/.) Most of the systems discussed in *Software Visualization* take the opposite approach, and try to use visualization to examine or explain existing programs. Some of the results leave me cold, but some, like the classic "Sorting Out Sorting" by Baecker et al., are as illuminating as Working Model's gears.

Of course, most contributions fall somewhere between these two extremes. The code-maintenance visualizer by Eick et al., for example, shows source files as strips, then color codes them line by line to indicate how frequently or recently they have changed. The resulting display would be a useful adjunct to a source-code control system, but its utility is limited by the fact that it reports changes to physical structure (lines), rather than logical structure (classes or methods). Still, like many other chapters in the book, it is an interesting starting point for further development, and provides a glimpse of what your next programming environment might look like.

Mark Murphy's *C/C++ Software Quality Tools* is a nice complement to this, since it looks at what your current programming environment ought to provide, but probably doesn't. According to the blurb on the back of this book, Murphy has worked on software quality assurance since 1990; judging from the book's clarity and common sense, I would have guessed that he'd been in the business much longer.

Along with three introductory chapters, two chapters on testing in general, and a brief concluding chapter, Murphy gives us one chapter on each of six different quality-assurance tools. The first of these is a home-brewed version of the *assert()* macro. This doesn't extend the functionality of the standard C++ *assert()* by much, but even experienced programmers will learn a few things from Murphy's discussion of when and how to use assertions.

The second tool he describes keeps track of dynamically allocated memory, and reports both out-of-bounds writes and memory leaks. Commercial varieties

---

Greg is the author of Practical Parallel Programming *(MIT Press, 1995), and coeditor with Paul Lu of* Parallel Programming Using C++ *(MIT Press, 1996). Greg can be reached at gvwilson@interlog.com.*

The Advertiser Index

Visit Our Advertisers Web sites Today!

Sales Director/West
Paul Miller
650-655-4183
pmiller@mfi.com

N. California/West
Stan Barnes
650-655-4190
sbarnes@mfi.com

Southwest/Texas
David Katch
650-655-4301
dkatch@mfi.com

Northwest
Michael Beasley
650-655-4304
mbeasley@mfi.com

Tom Hudner
650-655-4323
thudner@mfi.com

Fax: 650-358-9749

Sales Director/East
Brenner Fuller
603-746-3057
bfuller@mfi.com

Mid-Atlantic/Southeast
Nicole Glazen
781-235-8568
nglazen@mfi.com

Scott Rill
781-235-8577
srill@mfi.com

New England/ North Central
Linda Guyette
603-924-5971
lguyette@mfi.com

Software Careers/West
Hilary Walker
650-655-4193
hwalker@mfi.com

Software Careers/East
Stacey Mochizuki
650-655-4317
smochizuki@mfi.com

(continued from page 127)
of such tools, such as Purify (http://www.rational.com/products/purify/) and BoundsChecker (http://www.numega.com/ products/cc/screen_cb.htm), are taken for granted by most professional programmers, but this is the clearest explanation I have ever seen of how they work, and what they can and cannot do.

The third, fourth, and fifth chapters on tools describe an automatic test-case generator, a scripting language that can be used to drive tests, and a coverage tool that keeps track of which statements and branches in a program have been executed. All of these ought to be as widely known and widely used as the memory checkers discussed in the previous paragraphs; but because testing is a low-status occupation, they are ignored by most educators and professionals (see http://www.ddj.com/oped/1997/wilson.htm).

The last of Murphy's tool-oriented chapters describes a tool for capturing and replaying keystrokes and mouse movements, so that GUI testing can also be automated. Some people question the effectiveness of automated testing (see http://www.stlabs.com/testnet/docs/snakeoil.htm, for instance), but the tool itself is still worthy of study.

As good as it is, *C/C++ Software Quality Tools* does have one major weakness, which is its use of a home-grown scripting language as a test driver. Several well-documented, extensible, general-purpose scripting languages are widely used today, including Perl, Tcl, and Python. Any of these would, in my opinion, have been a better choice for Murphy than developing his own. What's more, the test generator and coverage tool would both be easier to extend or retarget if they had been written in one of these higher-level languages, rather than in C or C++. Despite this, and a lack of exercises and sample problems, this book would be a solid text for an undergraduate course on quality assurance.

Speaking of Perl...

I've been working my way through three new books on the language during the past few weeks, and have been impressed by all of them. Nigel Chapman's *Perl: The Programmer's Companion* and Jon Orwant's *Perl 5 Interactive Course* are both tutorials. At nearly 300 pages instead of Orwant's back-breaking 900, Chapman's book is thinner and more discursive, but contains neither self-quizzes nor exercises. Orwant's book, in contrast, describes Perl a snippet at a time, and has hundreds of multiple-choice questions and exercises. Orwant also goes into a lot more detail when discussing widely used Perl modules for process management, database access, and networking. I would recommend Chapman to Perl newcomers and Orwant to those who need to master the language, but given Chapman's lack of exercises, Orwant is probably a better buy either way.

Like Scott Meyers' *Effective C++*, on which it is modeled, Joseph Hall's *Effective Perl Programming* is not for the faint-hearted. Sixty items, grouped into 10 sections, cover everything from the difference between undefined and empty lists, through different ways of reading from a stream, to using the debugger effectively and submitting modules to the Comprehensive Perl Archive Network (http://www.perl.org/CPAN/, and many mirror sites). While the book is in some ways an indictment of Perl's quirkiness—if the language had fewer syntactic and semantic inconsistencies, the book could be a lot shorter—it ought to be in the stack underneath every serious Perl programmer's coffee mug.

The last of this month's books is Robert Glass's *Software Runaways: Monumental Software Disasters*. In this book, Glass looks at large projects that have spiralled out of control and subsequently failed. He states his thesis early:

> ...the most common problem in building software systems is not the construction of them itself, but rather the estimation of the costs of that construction. Why is there such a problem? Because the software field has not made a conscientious effort to develop histories of past project costs.

He then summarizes the findings of studies done in 1989 and 1995. Not surprisingly, failed projects tend to be those that were overly ambitious, for example, and project failure usually has several causes, rather than a single dominant cause. At the same time, using packaged software instead of rolling your own doesn't appear to reduce the risk of project failure, but using new technology (or technology that is new to the organization responsible for the project) greatly increases the risk.

If the bulk of the book was devoted to this sort of summarization and analysis, I would probably add it to my recommended reading list. Unfortunately, most of the rest of the book consists of reprints of articles about particular runaway projects, including the Denver airport baggage handling system, the FAA's next-generation Air Traffic Control system (which replaced single keystrokes with 12 or more mouse clicks), and Bank of America's MasterNet. Some of this is interesting, but only some, and only somewhat. Still, if it is not a step toward developing the history of past project costs that Glass says our profession needs, it is at least a signpost in that direction.

DDJ

Modern Software Publishing has released Modern GuardX, an application security solution for Windows 95/NT applications. GuardX acts like a firewall for applications, preventing unauthorized access to programs, program features, dialogs, and user-interface elements based on user logon credentials. GuardX lets you add U.S. Department of Defense C2-class security to business applications. GuardX supports Visual Basic, Access, Office, VBA, Visual C++, and other development environments. GuardX unit pricing is $698.00 with a one-year software maintenance subscription, free technical support, and a 30-day money-back guarantee.
Modern Software Publishing Inc.
Five Youngs Road
Pomfret Center, CT 06259
860-963-1071
http://www.modernsoftware.com/

ValiCert's recently announced ValiCert Toolkit 2.0 lets you embed certificate validation into any application. Version 2.0 of the toolkit supports all current digital certificate validation and revocation protocols, including mechanisms such as Certificate Revocation Lists (CRLs), the newly proposed Online Certificate Status Protocol (OCSP), and ValiCert's unique Certificate Revocation Trees (CRTs). The toolkit consists of software components that enable Internet applications to verify digital certificates using an array of established validation mechanisms. The ValiCert Toolkit 2.0 is available free of charge for an unlimited binary license and can be downloaded from http://www.valicert.com/.
ValiCert
3160 West Bayshore Road
Palo Alto, CA 94303
650-849-9860
http://www.valicert.com/

Plexstar has released Java modules that read, write, and display CAD files based on the DGN and DXF formats. The new modules are the DGN reader/writer add-on and the DXF reader/writer add-on.
Plexstar Inc.
3951 Pender Drive, Suite 105
Fairfax, VA 22030
703-246-9640
http://www.plexstar.com/

American Cybernetics has announced Version 8 of its programmer's text editor, Multi-Edit. Multi-Edit 8 features support for HTML, JavaScript, VBScript, Perl, Java, and other languages. Multi-Edit includes Project and Site Management tools, and IDE integration with Borland Delphi and C++ Builder. A free, fully functional demo of Multi-Edit 8 is available for download at http://www.multiedit.com/.
American Cybernetics Inc.
1830 West University Drive, Suite 112
Tempe, AZ 85281
602-968-1945
http://www.multiedit.com/

Software Tree has released JDX 1.0, an all Java implementation that provides an object-oriented interface to store and retrieve Java objects in a relational database. JDX provides support for complex object modeling and for object integration with legacy data.
Software Tree Inc.
650 Saratoga Avenue
San Jose, CA 95129
408-557-6769
http://www.softwaretree.com/

Formula One/Java from Visual Components is an Excel-compatible spreadsheet written in Java. Formula One/Java provides Excel read-write compatibility and more than 130 worksheet functions. Formula One/Java can be downloaded from http://www.f1j.com/. A developer license can be purchased for $49.00 per developer. Server licenses are available for $499.00 per server/per application. An unlimited distribution license is available for $5000.00.
Visual Components Inc.
12980 Metcalf Avenue, Suite 300
Overland Park, KS 66213
800-884-8665
http://www.visualcomp.com/

Albert's Ambry has introduced Zeus, a text editor/IDE designed specifically for programmers. Zeus is customizable, language neutral, and works seamlessly with all third-party development tools and compilers. Zeus is available in 16- and 32-bit versions, both of which support background compiling and build task runs. The editor supports keyboard mapping for Brief, WordStar, Epsilon, and Emacs, and has a built-in macro scripting language and color syntax highlighting. Zeus sells for $95.00 for a single-user license (multiple licenses discounts available).
Albert's Ambry
P.O. Box 2092
Bala Cynwyd, PA 19004
610-623-2014
http://www.alberts.com/authorpages/00002366/prod_270.htm

The Open Group has announced X11R6.4, the latest release of the X Window System technology. This release of the X Window System technology allows existing shrink-wrapped or custom applications to be accessed by an X11-enabled Web browser through such features as universal access and Low-Bandwidth X. New extensions and features include a reduction in the X server's memory footprint, Easy Resource Configuration (ERC), the ability to allow a multithreaded system to function as one large screen, Low-Bandwidth X, Display Power Management Signaling, and Colormap Utilization Policy.
The Open Group
11 Cambridge Center
Cambridge, MA 02142
617-621-8700
http://www.opengroup.org/

Micro Digital and Swell Software have announced PEG, a GUI for embedded systems. PEG provides a set of graphical objects such as windows, status and menu board, menus, buttons, icons, sliders, thumbwheels, tables, boxes, and more. PEG was designed specifically for embedded systems; the code size is under 100 KB. Drivers for VGA, SVGA, and LCD controllers are included, as well as support for mouse, touch screen, and keyboard input. The PEG Font Capture utility converts standard font files to the PEG format. The PEG Image Convert utility converts PCX, BMP, and TGA files to the compressed PEG format. PEG is supplied as a C++ library with source code, and costs $5000.00 royalty-free for one developed product.
Micro Digital Inc.
12842 Valley View Street, #208
Garden Grove, CA 92845
800-366-2491
http://www.smxinfo.com/

Metrowerks now officially supports the GNU GCC compiler from within the CodeWarrior environment (as an alternative to Metrowerks' own compiler). A subsidiary, Quorum Technologies, has been formed for the express purpose of supporting GCC within CodeWarrior. Quorum Technologies will provide bug fixes and full-service support for the GNU com-

pilers, linkers, and debuggers. It will build reference versions of GCC compilers and linkers that are validated against Metrowerks CodeWarrior GCC adapter technology.
Metrowerks Inc.
9801 Metric Boulevard, Suite 100
Austin, TX 78758
512-873-4700
http://www.metrowerks.com/

ParaGraph PI, the Pen and Internet Technology business unit of Silicon Graphics, has announced the CalliGrapher 5.1 handwriting-recognition tool for Windows CE. Features in CalliGrapher 5.1 include note-taking and screen draw support, support for color ink on color-enabled handheld PCs, a new method of quickly entering punctuation via pop-up menus, and an enhanced screen orientation feature.
ParaGraph PI
2011 N. Shoreline Boulevard, Bldg. 10
Mountain View, CA 94043
650-933-3000
http://www.paragraph.com/

ObjectAda Real-Time RAVEN from Aonix employs a fast real-time kernel that is suited for hard real-time and safety-critical applications. RAVEN's design is based on the Ada95 tasking restrictions set named the RAVENSCAR Profile, which accommodates certification requirements for high-integrity (safety-critical), real-time systems. The profile defines a special Ada95 tasking model with special emphasis on small size, fast performance, and deterministic behavior. RAVEN supports Solaris, HP/UX, and Windows NT. Target support for RAVEN includes Motorola's PowerPC, MC680x0, and MC683xx. Pricing starts at $15,000.
Aonix
595 Market Street, 12th Floor
San Francisco, CA 94105
415-543-0900
http://www.aonix.com/

ActiveState has released ActiveState Perl Debugger, a visual debugging environment for Win32 Perl programmers. A single license costs $95.00; multiple licenses are available at discounted rates.
ActiveState Tool Corp.
P.O. Box 2870 Main Station
Vancouver, BC
Canada V6B 3X4
604-606-4686
http://www.activestate.com/

Dynamical Systems has introduced readyBase, a dynamic and embeddable relational database library with a high-level C API, as well as Perl and Java interfaces. Features include rapid schema prototyping, a fast and flexible engine, powerful query language, and platform independence. readyBase is currently available for Linux, Solaris, and Irix.
Dynamical Systems Ltd
175 W. Jackson Boulevard, Suite A253
Chicago, IL 60604
312-341-1755
http://www.dynamical-systems.com/

Visual LISP from Autodesk is an integrated development environment that supports an extension of the AutoLISP programming language. Visual LISP supports compiled LISP, enhanced performance when running through AutoCAD Release 14, and an ActiveX interface.
Autodesk Inc.
111 McInnis Parkway
San Rafael, CA 94903
415-507-5000
http://www.autodesk.com/

Artisan Software Tools has announced Version 2.0 of its Real-Time Studio modeling environment for developing real-time systems. Version 2.0 features include UML-based requirements modeling, UML-based solution design, enhanced round-trip engineering, timing notes support, and a multiuser architecture. Real-Time Studio 2.0 runs on Windows 95/NT, and supports C++ and Java. A single user license costs $4995.00.
Artisan Software Tools Inc.
820 Bay Avenue, Suite 120
Capitola, CA 95010
408-475-5554
http://www.artisansw.com/

Halcyon Software has introduced Instant Basic for Java, a visual development environment compatible with Microsoft Visual Basic. Instant Basic for Java includes a compiler that compiles VB source code directly to Java bytecode. You can debug Java applications at the VB source-code level. The Professional Edition includes Professional Controls, ActiveX support, and DAO/RDO support with JDBC. Instant Basic for Java Standard Edition costs $99.00, while the Professional Edition costs $795.00.
Halcyon Software
1590 La Pradera Drive
Campbell, CA 95008
408-378-9898
http://www.halcyonsoft.com/

ObjectStore PSE Pro Release 2.0, from Object Design, is a Java object database. ObjectStore PSE Pro stores and retrieves Java objects without translation, offers full querying and indexing capabilities, and supports JDK 1.2 collections. ObjectStore PSE Pro costs $245.00 per developer; an end-user license costs $95.00 per user with quantity discounts available.
Object Design Inc.
25 Mall Road
Burlington, MA 01803
781-674-5000
http://www.objectdesign.com/

APEX Software has announced Version 5.0 of its True DBGrid Pro ActiveX grid control. Version 5.0 includes multiple lines per record, data-sensitive colors and graphics, drop-down data-aware list boxes, input masking, and context-sensitive CellTips. Pro 5.0 also includes reusable grid layouts and an add-in grid Design Assistant. TrueDBGrid Pro 5.0 costs $399.95.
APEX Software Corp
4516 Henry Street
Pittsburgh, PA 15213
412-681-4343
http://www.apexsc.com/

The KL Group has announced JProbe Profiler, a graphical hierarchical profiler for Java applets and applications. At its core is a standard Java VM, licensed from JavaSoft and instrumented by KL Group. Features include the ability to graphically view memory usage and instance counts of each object as the Java application runs, a Call Graph Interface, and an advanced source code display. JProbe runs on Windows NT and costs $499.00.
KL Group Inc.
260 King Street East
Toronto, ON
Canada M5A 1K3
416-594-1026
http://www.klg.com/

POET Software announced its new POET Content Management Suite, an extensible application built upon the POET object server that adds XML/SGML functionality to accelerate deployment of solutions for structured content management. The POET Content Management Suite includes the POET Content Client, the POET Content Server, and the POET Content SDK. In addition to its support for XML/SGML content, it supports full-featured revision control, full-text indexing, workgroup collaboration, and previous version rollback. The POET Content SDK includes both a C++ and Java API. Pricing is $1500.00 for a client license, and $20,000 for a complete end-user package, including training and support.
POET Software Corp.
999 Baker Way, Suite 100
San Mateo, CA 94404
650-286-4640
http://www.poet.com/

DDJ

DR. DOBB'S JOURNAL

PRODUCT DEBUT

DR. DOBB'S JOURNAL

MARKETPLACE

DOCUMENTATION TOOLS

C++ to HTML

- New - Version 2.1.
- Generates documentation directly from the source code.
- Extracts comments.
- User customized reports formats.
- HTML, WinHelp, RTF.
- FREE working evaluation at www.bbeesoft.com

1-888-646-1933
Bumble Bee Software
P.O. Box 2007
Westford, MA 01886
info@bbeesoft.com

GRAPHICS LIBRARIES

SciTech MGL
Now FREE with Full Source Code

SciTech MGL is a complete graphics library for Windows 95/3.1/NT & DOS that has been used to develop leading titles like *WinQuake®* and *Hexen II®*. SciTech MGL can be ported to other OS's in as few as 1000 lines of code. **Includes:** OpenGL® API support; sprite library; Game Framework; hardware triple buffering; support for stereo LCD shutter glasses; automatic detection & utilization of VGA, ModeX, VESA VBE, VBE/AF, WinG, CreateDIBSection & DirectDraw; and more. Supports standard C/C++ compilers & Borland Delphi.
For more information call **(530) 894-8400.**
Download SciTech MGL with full source code from our website at:
www.scitechsoft.com

SECURITY

Securely Sell Your Software Worldwide

Rainbow has software, hardware and Internet solutions to help developers

- *Increase Sales*
- *Expand Distribution*
- *Prevent Piracy*

800-705-5552
www.rainbow.com
RAINBOW TECHNOLOGIES *Securing Software Success*

EDUCATION

Netscape: Student Section

ADDRESS FOR SUCCESS
http://www.aics.edu

- Earn B.S. and M.S. in Computer Science
- DISTANCE EDUCATION
- Object oriented B.S. program
- New courses in Java, Networking, HTML, MIS
- Approved by more than 275 companies
- Follows ACM/IEEE guidelines
- Thousands of students throughout U.S.

AMERICAN INSTITUTE COMPUTER SCIENCES
ACCREDITED MEMBER World Association of Universities and Colleges

Free catalogue **1-800-767-AICS**
or **http://www.aics.edu.**

PRODUCTS WANTED

WANTED
SOFTWARE PROGRAMS
LOOking for partners in profit!

You provide the completed Small Business, SOHO, Financial or Personal Productivity software program. E-Z Legal Software will provide packaging, duplication, marketing and sales personnel to generate royalty checks for you.

With distribution in more than 6500 retail outlets, we have the experience to launch, market and sell your program at retail at no cost to you! **LINDA ZACK: 1-800-822-4566 x131**

384 S. Military Trail, Deerfield Beach, FL 33442
Phone (954) 480-8933 • Fax (954) 480-8906
http://www.e-zlegal.com — lzack@e-zlegal.com

E-Z LEGAL SOFTWARE

Reliable Software Security

Cost effective products for all levels of software protection and authentication.

- CRYPTO-BOX® brand dongles, SmartCard, PC Cards, and extension cards
- Software based metering systems
- ActiveX, OCX & VCL components

VISIT US www.marx.com ON THE WEB

MARX® SOFTWARE SECURITY
770-986-8887 FAX: 770-986-8891
1-800-627-9468
2900 Chamblee-Tucker Rd., Bldg. 9, Atlanta, GA 30341-4151

ELECTRONIC FORMS

Add ANY FORM To Your Windows Application!

Speed **up your software development time by using Nelco's extensive form libraries & form creation services.**

• PCL, Placeable Metafiles, PDF Formats • Field Format Information for Every Form • Sample Metafile Code in Visual Basic & Delphi • Government Approved Forms • Scan/barcoding Abilities

Visit our Software Developer's page:
http://www.taxforms.com/A.html
Call: 1-920-337-1013
E-mail: nelcomktg@nelcoinc.com.

Nelco®
PO Box 10208
Green Bay WI 54307

PROLOG TOOLS

Add Intelligent Components

*Diagnose, advise, configure and plan with the Amzi!® LogicServer™ tools & libraries (DLLs) for C/C++, Java, VB, Delphi, Web Servers & more. Win NT 95 3.x & Solaris. Use ODBC, Sockets, Unicode & **new** OOP extensions.*

FREE Evaluation Version!

Amzi! inc. *Email* info@amzi.com
Call +1-513-425-8050 *Fax* 425-8025

www.amzi.com

TOOLS

Understand For C++

Analyzes your C++ source code to help you quickly understand your software.

- Hypertext source & Class Browser
- Detailed Interactive Cross Reference
- Invocation, Include, Declaration Heirarchy Charts
- HTML Publishing of Code & Analysis
- WYSIWYG printing, Clipboard & Bitmap Output

Download your eval copy today.
www.scitools.com
(603) 448-6960

STI
Scientific Toolworks, Inc.
115 Etna Road Suite 18A
Lebanon, NH 03766

Understand Your Software...

GRAPHICS LIBRARIES

☑ GUI

- Portable Embedded GUI
- Small, fast, easily ported C++ library
- VGA, LCD, and input drivers included

For full information and free demo:
www.smxinfo.com/pegdd.htm

MICRO DIGITAL INC 800-366-2491
Innovation Leader 714-373-6862

SCRIPTING LANGUAGES

CGI scripting is a piece of cake with Winductor 2.1

- Easy and clear syntax
- Native to Windows
- Integrated debugging — can be debugged with actual POSTed data from browser
- Use e-mail, FTP, HTTP, HTML-parsing and file I/O directly from the scripts
- Script and orchestrate other Windows programs
- Easy-to-follow CGI tutorial

Download your free evaluation copy today
Winductor 2.1

www.desisoft.com Desiderata Software
2 Vernon St.#551, Framingham, MA 01701

C and C++ DOCUMENTATION TOOLS (v. 7.0)

- **C-CALL ($69)** Graphic-tree of caller/called function hierarchy, cross-reference, file/function index.
- **C-CMT ($69)** Creates/inserts/updates comment-blocks (functions/identifiers used) for each function.
- **C-METRIC ($59)** Calculates path complexity, counts lines with comments, code, 'C' statements.
- **C-LIST ($69)** Lists and action-diagrams, or reformats source into user-selected standard formats.
- **C-REF ($69)** Creates cross-reference of local/global/define/parameter identifiers.
- **C-DOC ($199) PACKAGE** All 5 programs integrated as DOS program. <10,000 lines. C-BROWSE Windows graphic-tree viewer.
- **C-DOC Professional ($299)** DOS, Windows, OS/2, 1,000,000+ lines.
- **NEW VER 7.0! WEB HTML REPORTS!**

SOFTWARE BLACKSMITHS INC. email @ swbs.com
6064 St Ives Way, Mississauga Voice/Fax **(905) 858-4466**
ONT Canada L5N-4M1 http://www.swbs.com

★★★★★
VICTOR
Image Processing Library

Fast **BMP, TIFF, PCX, GIF, TGA, PNG, JPEG.** Adjust brightness, contrast, sharpen, create filters, resize, rotate, +more of single image, multiple images, or any image area; color reduction to optimum, specific, or std. palette; print; scan; crop, combine, compare, blend images.

DOS $199, 16-bit DLL $299, 32-bit DLL $499

Catenary Systems
314-962-7833/fax: 314-962-8037
www.catenary.com/victor
ask for free demo src avail visa/mc/c.o.d.

SECURITY

CrypKey Software Licensing System
"Software protection with **NO** hardware lock and **NO** disk key"

CrypKey **is software copy protection** that is:
- completely secure from any disk copy program
- perfect for **CD-ROM** or **INTERNET** distribution!
- cost effective, user friendly, and **100% guaranteed** to satisfy!

CrypKey can **increase your software sales:**
- **upsell options and levels** of your software
- **lease or demo** your software by runs or time
- **enable or upgrade** your customers **instantly by phone, fax or email!**

New! unique **Ready-To-Try** features upon install allows 1 trial period only per customer.
New! unique **Add-On** feature-add more options, levels, runs or time to existing licenses.
New! CrypKey Instant-protects in just **5 minutes** with no source code changes.

CrypKey is completely compatible with MS-DOS, MS-Windows 3.x, Win32s, Win95, Win NT, and manages network licenses on all Novell and Microsoft operating system based networks. CrypKey is produced by Kenonic Controls Ltd.—engineering and software since 1972.

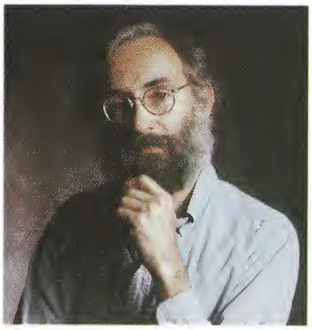
Kenonic Controls Limited
7175 - 12th Street South East
Calgary, Alberta, Canada T2H 2S6
(403) 258-6200 fax: (403) 258-6201
INTERNET: crypkey@kenonic.com
WEB: http://www.kenonic.com/crypkey.html

Java & C++ Training
for Developers Who Need to Deliver

- Ultimate Java Programming
- Advanced Java Programming
- C++ Programming
- Object Oriented Analysis & Design

Now offering JavaBeans Java 1.2 & JFC

PETRONIO
Technology Group
781-778-2000
FREE Java Quick Reference Card
www.petronio.com/refddj



The Dublin Corps

I was loosening tongues with alcohol, working the late bartending shift at Foo Bar, the Silicon Valley hangout where the elite meet to be indiscreet. I had just poured British journalist Laurence Wilde his first glass of Chardonnay of the evening and refilled Maureen McBean's Haig & Haig. Joe Weaver hadn't come in yet.

Joe was under no obligation to hang out with Larry and Mo, I reminded myself, but I was so used to seeing the three journalists together that I had come to think of them as the Three Stooges. When one of them was tardy, there was a certain ineffable *je ne sais quoi* missing from the eye gouging and the head thumping.

"This might interest you, Mo," Larry said with lamentable civility. (Lamentable from my point of view, that is. I only moonlight at Foo Bar for the entertainment value.)

Just then, Joe walked in. My spirits rose, but fell immediately. Clamped between his teeth was an enormous cigar. Foo Bar is a no-smoking establishment. I'd have to put on my enforcer hat.

"What's with the stogie, Joe?" Mo asked, swiveling on her bar stool.

He sat and held the thing out admiringly between two fingers. "All the cool people are into cigars."

I plunked his customary cream soda down in front of him. "Joe—"

But he was on a roll. "Cigar parlors are the hot thing now. Tom Cruise and Arnold Schwartzeneg..."

Larry pushed the peanut dish in Joe's direction. "Positively fascinating, Joe. By the way, you might find this interesting—"

"...are big cigar smokers. I don't mean they smoke big cigars, although they might...."

I tried to cut in. "You know, Joe, we have a policy —"

"I've been asked to participate in the Dublin Core evaluation," Larry went on doggedly, intent on getting his point across. Joe turned toward me, smiling blandly, but Larry had caught Mo's attention.

As I wrangled with Joe over the cigar issue, I half-listened to Larry and Mo. From what I could hear, they were no more than half-listening to each other.

"The Dublin Corps, huh?" Mo said. "You know my family is Irish."

"There's quite an alphabet soup of schemes for facilitating search in electronic documents—XML, XSL, RDF. It's positively dizzying keeping them straight. The Dublin Core is just one of them."

"A Brit would say that. It's not just one of them. It's the most important one. It's our island."

Larry pressed on. I was explaining to Joe that there is no cigar exemption to the California smoking ban, whatever he may have heard. "But the Dublin Core initiative," Larry said, "has already produced international consensus on a base set of 15 elements for descriptive metadata."

Mo pushed her glass at me and mimed refilling it. "Okay, George Mitchell has done wonderful work with the peace negotiations, but, Larry, you can't expect people to forget centuries of oppression. You Brits have got to get that through your heads."

"And it appears that the big guns at W3C are receptive to the DC proposals."

"Guns in the WC? You Limeys are such hoodlums."

"The 15 elements are already more or less supported in HTML 4. That is, meta tags in HTML 4 are underspecified. You clarify them by including a reference to a profile, a URL of a document that defines useful properties for meta tags. Dublin Core is one such profile."

Mo stared at him with the expression of a woman who has just awakened and found a reference librarian in her bed. "What the heck are you talking about?"

"The Dublin Core. The 15 metadata elements. Title, subject, description, source, language, relation, coverage, creator, publisher, contributor, rights, date, type, format, and identifier." She continued to stare at him for a moment and then turned to Joe. "Joe, you can't smoke that in here," she said, and took the cigar and crushed it on a coaster.

"But I wasn't going to smoke it," Joe wailed. "I just have it because cigars are the in thing."

The heck with these clowns. I'm going to run off and join the Dublin Corps. You can, too, at http://purl.org/metadata/dublin_core/.

Michael Swaine

Michael Swaine
editor-at-large
mswaine@swaine.com

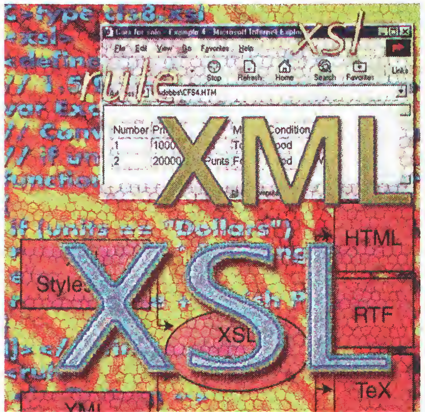
Rendering XML Documents Using XSL

Keeping content and format separate

Sean McGrath

Central to the eXtensible Markup Language (XML) philosophy is that the structure and content of information should be captured without concern for how the information will be rendered on a computer display, paper, voice synthesis, and others. Responsibility for rendering XML has been delegated to a sister standard known as eXtensible Style Language (XSL). (For more information on XML, see my article "XML Programming in Python," *DDJ*, February 1998.)

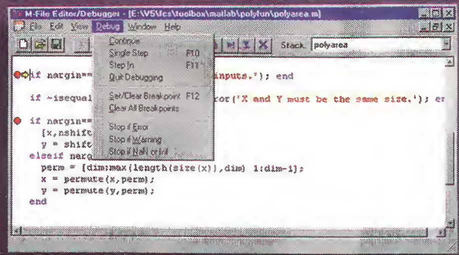
Like XML, XSL is a World Wide Web Consortium (W3C) initiative. In August of 1997, a draft proposal for XSL was made .html). Although, the working draft for XSL is just that, a number of XSL applications have already appeared. In particular, Microsoft has released MSXSL, a "technology preview" implementation that is freely available at http://www.microsoft.com/xml/. In this article, I will present an overview of XSL and illustrate how it can be used with MSXSL.



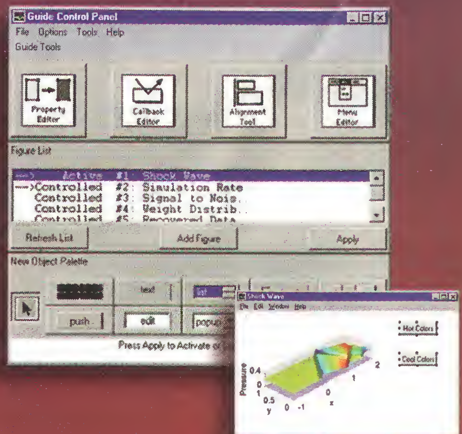
pens late — preferably right at the point that someone wants to view it (hence, late binding).

This late binding approach has some significant benefits:

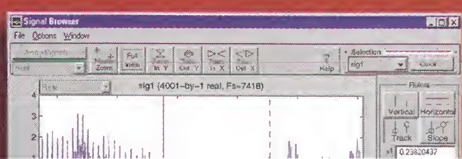
- The look and feel of a document (or thousands of documents) can easily be changed simply by changing the stylesheet.
- Multiple renditions of the same content can be created from a single source. These renditions can include different output notations such as RTF, HTML, or Postscript. They can involve rearrangements of the content, creating multiple views of the information.
- The information content is "future proofed." Creating a new rendition to a new notation (or a notation yet to be invented), is a matter of applying the necessary stylesheet.
- Keeping the content free of rendering information makes it easier to process the content. That is, searching, harvest-



Application Development Tools
"We initially coded algorithms in MATLAB and then converted the MATLAB source to C or C++. To our surprise, the MATLAB code was faster in nearly all cases."
Jack Staub
Hughes Aircraft



Interactive GUI Design
"In one day, I wrote 875 lines of MATLAB which equates to 5,000 lines of C code. I had a functioning GUI in one day. You can't do that with C."
Kathleen Splaine
Risk International



We wrote exactly 698,794 lines of C code so that you don't have to.

More than 400,000 engineers and scientists use MATLAB to accelerate their technical programming. Here's why.

Faster programming
Today's most productive technical professionals have one thing in common – they use MATLAB instead of C or C++. Because, unlike a general purpose language, MATLAB is a complete, integrated analysis, visualization, modeling, and development environment specifically designed for technical computing. So development goes much faster and code is dramatically shorter.

More numerical power built in
At the heart of MATLAB is an easy to learn technical computing language with more than 500 functions built in. The MATLAB language includes flow control, multidimensional arrays, user-defined structures, ASCII and binary external data file access, and much more. And you'll save even more time with MATLAB Toolboxes, collections of highly optimized, discipline-specific functions written by world-class experts. You can even link in C, C++, and Fortran routines.

MATLAB 5
COMPILER
NOW AVAILABLE

Less time coding means more time to think
Put simply, use MATLAB and it will take you far less time to develop finely tuned applications with revealing graphics, custom GUIs, and compact maintainable code. Now just imagine what you can do with the time you save.

See how MATLAB 5 can help you work faster. Visit our Web site for demos, examples, and updating information.

With JDK 1.1-based browsers, this restriction does not apply if the applet is signed.

Oracle Connection Manager may be deployed in combination with JDBC applets to provide secure access to Oracle environments. Connection Manager incorporates a Net8 application proxy, which lets system administrators control how a connection request gets routed. Through the use of rules, requests may be filtered based on parameters such as:

- Destination or Origin IP address.
- Oracle System Identifier (SID).
- Data encryption/security preferences.

The ThinJDBC applet can connect to a Connection Manager running on the webserver host and have the Connection Manager redirect the packets to an Oracle server running on a separate host.

Fast Forward

FastForward, a Type IV driver from Connect Software, provides Java clients with direct access to Microsoft SQL Server (all versions) and Sybase SQL Servers (Versions 4, 9, 10, and 11). FastForward works by directly transferring and receiving information from Java to SQL Server using TCP/IP sockets. The format of data passed back and forth is TDS. Version 3.0 offers features such as xencryption and HTTP tunneling through the FastForward Security Proxy. FastForward Proxy is a pure Java application that provides connectivity between clients on the Internet and servers within your network. It also provides HTTP tunneling, compression, and encryption. Proxy uses symmetric private key encryption in 16 rounds with 64-bit key for encryption. It also supports DES.

OPENjdbc

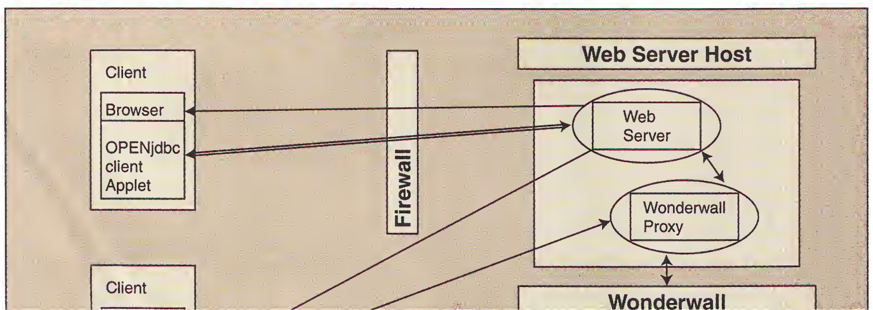
OPENjdbc is I-Kinetics CORBA-based Type III driver. The driver communicates to the DataBroker server through IIOP protocol. The databroker server is based on CORBA, using Iona's Orbix Object Request Broker to provide features such as multithreading, connection pooling, and load balancing. OPENjdbc invokes methods and services defined in the DataBroker's IDL files. Figure 5 illustrates the OPENjdbc architecture.

OPENjdbc driver offers security features such as SSL encryption and authentication (through Orbix SSL) and HTTP tunneling of IIOP. The SSL option is provided through Orbix SSL, which allows Orbix- and Orbix-Web-based applications to be easily retrofitted with SSL security. Orbix SSL replaces the default IIOP protocol with the standardized SSL-IIOP protocol, which is essentially IIOP over secure SSL connections. The SSL option provides authentication using public-key cryptography (RSA, DSS) and encryption using block-encryption methods (DES, RC4).

The DataBroker server need not be on the same host as the web server that served the applet. An IIOP proxy on the web server can route all requests to the databroker. Iona's Wonderwall offers this feature along with its firewall features such as examination, filtering and logging of IIOP requests, HTTP tunneling support, and ACL support.

Conclusion

Security is an important aspect of applications that deal with sensitive data and are deployed on an open medium such as the Internet or an open intranet (no firewall). JDBC driver vendors offer tools which could meet the security needs of these applications. Your choice of software should be made based on factors such as application requirements, deployment configuration, security needs, expandibility, and standards supported by the vendor.



For More Information

WebLogic Inc.
417 Montgomery Street
San Francisco, CA 94104
415-659-2600
http://www.weblogic.com/

Intersolv Inc.
9420 Key West Avenue
Rockville, MD 20850
301-838-5000
http://www.intersolv.com/

Caribou Lake Software Inc.
4780 Beacon Hill Road
St. Paul, MN 55122
612-688-9470
http://www.cariboulake.com/

SCO
425 Encinal Street
P.O. Box 1900
Santa Cruz, CA 95061-1900
408-425-7222
http://www.sco.com/

Sybase Inc.
6475 Christie Avenue
Emeryville, CA 94608
510-922-3555
http://www.sybase.com/

Oracle Corp.
500 Oracle Parkway
Redwood Shores, CA 94065
650-506-7000
http://www.oracle.com/

Connect Software Inc.
81 Lansing Street

OPENGL

Listing One

```
typedef struct
{
    float x, y, z;  /* coordinates */
} make_vertex_list;
```

Listing Two

```
typedef struct
{
    int a, b, c;    /* array indices */
    int a_s, a_t,   /* (s, t) texture coordinates */
    b_s, b_t,
    c_s, c_t;
} make_index_list;
```

Listing Three

(a)

```
(vertex_list[index_list[0].a].x, vertex_list[index_list[0].a].y,
                                  vertex_list[index_list[0].a].z)
```

(b)

```
(vertex_list[index_list[0].b].x, vertex_list[index_list[0].b].y,
                                  vertex_list[index_list[0].b].z)
```

(c)

```
(vertex_list[index_list[0].c].x, vertex_list[index_list[0].c].y,
                                  vertex_list[index_list[0].c].z)
```

Listing Four

```
typedef struct
{
    make_vertex_list *vertex;
} make_frame_list;
```

Listing Five:

(a)

```
frame_list[F].vertex[index_list[P].a].x
frame_list[F].vertex[index_list[P].a].y
frame_list[F].vertex[index_list[P].a].z
```

(b)

```
frame_list[F].vertex[index_list[P].b].x
frame_list[F].vertex[index_list[P].b].y
frame_list[F].vertex[index_list[P].b].z
```

(c)

```
frame_list[F].vertex[index_list[P].c].x
frame_list[F].vertex[index_list[P].c].y
frame_list[F].vertex[index_list[P].c].z
```

Listing Six

(a)

```
R: m_palette_buffer [ m_pixel_buffer[0]]
G: m_palette_buffer [ m_pixel_buffer[1]]
B: m_palette_buffer [ m_pixel_buffer[2]]
```

(b)

```
R: m_palette_buffer [3 * m_pixel_buffer[P]+0]
G: m_palette_buffer [3 * m_pixel_buffer[P]+1]
B: m_palette_buffer [3 * m_pixel_buffer[P]+2]
```

(c)

```
R: m_palette_buffer [3 * m_pixel_buffer[X + Y*Width]+0]
G: m_palette_buffer [3 * m_pixel_buffer[X + Y*Width]+1]
B: m_palette_buffer [3 * m_pixel_buffer[X + Y*Width]+2]
```

Listing Seven

```
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

Listing Eight

```
glEnable(GL_CULL_FACE);
glEnable(GL_TEXTURE_2D);
glPolygonMode (GL_FRONT, GL_FILL);
/* use the OpenGL function to rescale */
gluScaleImage (GL_RGBA, m_iWidth, m_iHeight, GL_UNSIGNED_BYTE, unScaled,
          m_iscaledWidth, m_iscaledHeight, GL_UNSIGNED_BYTE, glTexture);

/* reclaim memory of the unscaled texture */
delete [] unScaled;
```

TEXTURE MAPPING

Listing One

```
void Draw_Triangle(float x0,float y0,float x1,float y1,
                                    float x2,float y2, int color)
{
// this function rasterizes a triangle with a flat bottom

// compute left side interpolant
float dx_left = (x2 - x0)/(y2 - y0);

// compute right side interpolant
float dx_right = (x1 - x0)/(y2 - y0);

// seed left and right hand interpolators
float x_left = x0;
float x_right = x0;

// enter into rasterization loop
for (int y=y0; y<=y1; y++)
     {
      // draw the scanline
      Draw_Line(x_left, x_right, y, color);

     // advance interpolants
     x_left+=dx_left;
     x_right+=dx_right;

     } // end for y
} // end Draw_Triangle
```

Listing Two

```
// initialize u,v interpolants to left and right side values
ui = ul;
vi = vl;

// now interpolate from left to right, i.e, in a positive x direction
for (x = xstart; x <= xend; x++)
     {
      // get texture pixel value
      pixel = texture_map[ui][vi];

      // plot pixel at x,y
      Plot_Pixel(x,y,pixel);

      // advance u,v interpolants
      ui+=du;
      vi+=dv;
      } // end for x
```

DVD

Listing One

```
// grab an interface to the Annex J methods
hr = m_pgraph->QueryInterface(IID_IDvdControl, (void **) &m_pUserOperations);

if ( !(FAILED(hr)) )
{
    // start playing title 1, chapter 3
    hr = m_pUserOperations->ChapterPlay( 1, 3 );

    // view the 2nd angle
    m_pUserOperations->AngleChange( 2);

    // turn off annoying foreign language subtitles
    m_pUserOperations->SubpictureStreamChange( 1, FALSE );

    // release interface
    m_pUserOperations->Release();
}
```

Listing Two

```
switch (Event)
```

Rendering XML Documents Using XSL

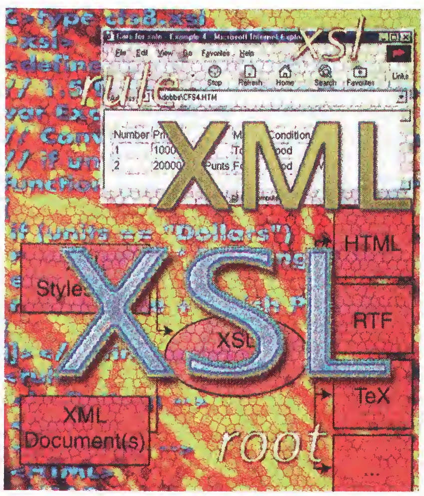
Keeping content and format separate

Sean McGrath

Central to the eXtensible Markup Language (XML) philosophy is that the structure and content of information should be captured without concern for how the information will be rendered on a computer display, paper, voice synthesis, and others. Responsibility for rendering XML has been delegated to a sister standard known as eXtensible Style Language (XSL). (For more information on XML, see my article "XML Programming in Python," *DDJ*, February 1998.)

Like XML, XSL is a World Wide Web Consortium (W3C) initiative. In August of 1997, a draft proposal for XSL was made available as a discussion document by the W3C (http://www.w3.org/TR/NOTE-XSL.html). Although, the working draft for XSL is just that, a number of XSL applications have already appeared. In particular, Microsoft has released MSXSL, a "technology preview" implementation that is freely available at http://www.microsoft.com/xml/. In this article, I will present an overview of XSL and illustrate how it can be used with MSXSL.

Sean, chief technical officer and cofounder of Digitome Electronic Publishing (http://www.digitome.com/) is a member of the World Wide Web Consortium's XML Special Interest Group. He is also the author of ParseMe.1st: SGML for Software Developers *(Prentice-Hall, 1997) and* XML By Example: Building E-commerce Applications *(Prentice-Hall, 1998). Sean can be reached at sean@digitome.com.*



The XSL Philosophy

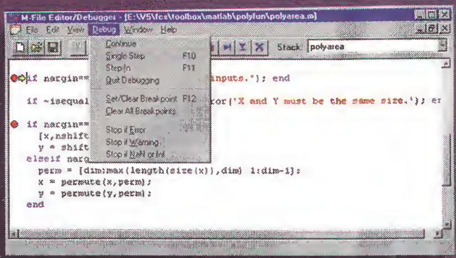
As Figure 1 illustrates, the XSL philosophy can be summed up as "late binding of presentation semantics." In simple English, the idea is that information about how a document should look when rendered (presentation semantics) is separated from the document content and housed in a stylesheet. The process of creating a rendition of the content happens late — preferably right at the point that someone wants to view it (hence, late binding).

This late binding approach has some significant benefits:

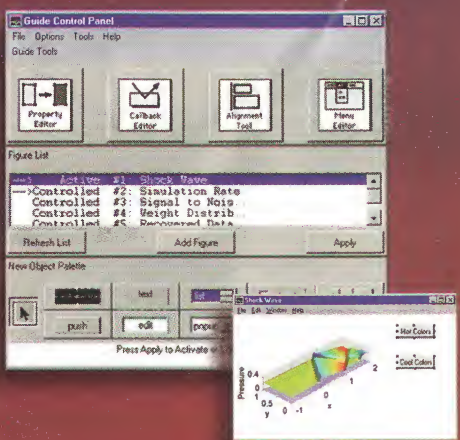
- The look and feel of a document (or thousands of documents) can easily be changed simply by changing the stylesheet.
- Multiple renditions of the same content can be created from a single source. These renditions can include different output notations such as RTF, HTML, or Postscript. They can involve rearrangements of the content, creating multiple views of the information.
- The information content is "future proofed." Creating a new rendition to a new notation (or a notation yet to be invented), is a matter of applying the necessary stylesheet.
- Keeping the content free of rendering information makes it easier to process the content. That is, searching, harvesting, or rearranging the content can be performed without worrying about how the formatting information is intermingled with the content.

There are a number of core concepts that are central to XSL, including:

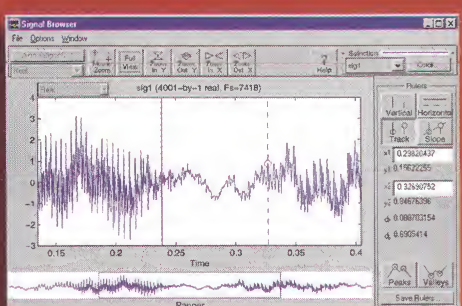
Flow Objects. In XSL, the process of transforming an XML document into a notation such as RTF, HTML, or Postscript, is expressed in terms of the construction of flow objects, which are



Application Development Tools
"We initially coded algorithms in MATLAB and then converted the MATLAB source to C or C++. To our surprise, the MATLAB code was faster in nearly all cases."
Jack Staub
Hughes Aircraft



Interactive GUI Design
"In one day, I wrote 875 lines of MATLAB which equates to 5,000 lines of C code. I had a functioning GUI in one day. You can't do that with C."
Kathleen Splaine
Risk International



Analysis and Visualization
"Anything from simple analysis to complex modeling and simulation can be done in a fraction of the time it would take to write your own code."
Gregory E. Chamitoff, Ph.D.
NASA, Johnson Space Center

We wrote exactly 698,794 lines of C code so that you don't have to.

More than 400,000 engineers and scientists use MATLAB to accelerate their technical programming. Here's why.

Faster programming

Today's most productive technical professionals have one thing in common – they use MATLAB instead of C or C++. Because, unlike a general purpose language, MATLAB is a complete, integrated analysis, visualization, modeling, and development environment specifically designed for technical computing. So development goes much faster and code is dramatically shorter.

More numerical power built in

At the heart of MATLAB is an easy to learn technical computing language with more than 500 functions built in. The MATLAB language includes flow control, multidimensional arrays, user-defined structures, ASCII and binary external data file access, and much more. And you'll save even more time with MATLAB Toolboxes, collections of highly optimized, discipline-specific functions written by world-class experts. You can even link in C, C++, and Fortran routines.

MATLAB 5 COMPILER NOW AVAILABLE

Less time coding means more time to think

Put simply, use MATLAB and it will take you far less time to develop finely tuned applications with revealing graphics, custom GUIs, and compact maintainable code. Now just imagine what you can do with the time you save.

See how MATLAB 5 can help you work faster. Visit our Web site for demos, examples, and updating information.

The MATH WORKS Inc.

MATLAB®
www.mathworks.com/ddjm
call 508-647-7000
e-mail info@mathworks.com

The MathWorks, Inc. 24 Prime Park Way, Natick, MA 01760 Fax 508-647-7001
Employment opportunities: http://www.mathworks.com/newjobs.html

The MathWorks is represented in the following countries: **Australia:** + 61-2-9922-6311 • **Benelux:** + 31(0)182-53-7644 • **Brazil:** + 55-11-816-3144 **Czech Republic:** 42-(0)2-68-44-174 • **France:** 33-141-14-67-14 • **Germany/Austria:** + 49-241-470750 • **India:** + 91-80-5-549338 **Israel:** + 972-3-561-5151 • **Italy:** + 39-11-240-80-00 • **Japan:** + 81-3-5978-5410 • **Korea:** + 82-2-556-1257 • **New Zealand:** + 64-7-839-9102 **Nordic Countries:** + 46-8-15-30-22 • **Poland:** + 48-126-17-33-48 • **Singapore/Malaysia:** + 65-842-4222 • **South Africa:** + 27-11-325-6238 **Spain/Portugal:** + 34(9)3-415-49-04 • **Switzerland:** + 41-31-954-2020 • **Taiwan:** 886-2-505-0525 • **United Kingdom/Ireland:** + 44-1223-423-200

With JDK 1.1-based browsers, this restriction does not apply if the applet is signed.

Oracle Connection Manager may be deployed in combination with JDBC applets to provide secure access to Oracle environments. Connection Manager incorporates a Net8 application proxy, which lets system administrators control how a connection request gets routed. Through the use of rules, requests may be filtered based on parameters such as:

- Destination or Origin IP address.
- Oracle System Identifier (SID).
- Data encryption/security preferences.

The ThinJDBC applet can connect to a Connection Manager running on the webserver host and have the Connection Manager redirect the packets to an Oracle server running on a separate host.

Fast Forward

FastForward, a Type IV driver from Connect Software, provides Java clients with direct access to Microsoft SQL Server (all versions) and Sybase SQL Servers (Versions 4, 9, 10, and 11). FastForward works by directly transferring and receiving information from Java to SQL Server using TCP/IP sockets. The format of data passed back and forth is TDS. Version 3.0 offers features such as xencryption and HTTP tunneling through the FastForward Security Proxy. FastForward Proxy is a pure Java application that provides connectivity between clients on the Internet and servers within your network. It also provides HTTP tunneling, compression, and encryption. Proxy uses symmetric private key encryption in 16 rounds with 64-bit key for encryption. It also supports DES.

OPENjdbc

OPENjdbc is I-Kinetics CORBA-based Type III driver. The driver communicates to the DataBroker server through IIOP protocol. The databroker server is based on CORBA, using Iona's Orbix Object Request Broker to provide features such as multithreading, connection pooling, and load balancing. OPENjdbc invokes methods and services defined in the DataBroker's IDL files. Figure 5 illustrates the OPENjdbc architecture.

OPENjdbc driver offers security features such as SSL encryption and authentication (through Orbix SSL) and HTTP tunneling of IIOP. The SSL option is provided through Orbix SSL, which allows Orbix- and Orbix-Web-based applications to be easily retrofitted with SSL security. Orbix SSL replaces the default IIOP protocol with the standardized SSL-IIOP protocol, which is essentially IIOP over secure SSL connections. The SSL option provides authentication using public-key cryptography (RSA, DSS) and encryption using block-encryption methods (DES, RC4).

The DataBroker server need not be on the same host as the web server that served the applet. An IIOP proxy on the web server can route all requests to the databroker. Iona's Wonderwall offers this feature along with its firewall features such as examination, filtering and logging of IIOP requests, HTTP tunneling support, and ACL support.

Conclusion

Security is an important aspect of applications that deal with sensitive data and are deployed on an open medium such as the Internet or an open intranet (no firewall). JDBC driver vendors offer tools which could meet the security needs of these applications. Your choice of software should be made based on factors such as application requirements, deployment configuration, security needs, expandibility, and standards supported by the vendor.

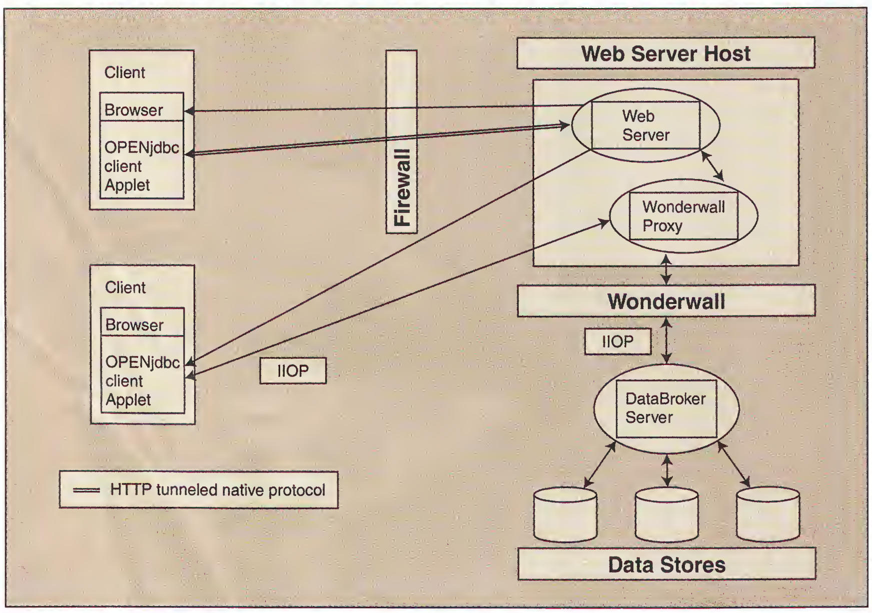


Figure 5: *The OPENjdbc client (client behind firewall, client outside a firewall) accessing DataBroker server where server is behind Wonderwall.*

For More Information

WebLogic Inc.
417 Montgomery Street
San Francisco, CA 94104
415-659-2600
http://www.weblogic.com/

Intersolv Inc.
9420 Key West Avenue
Rockville, MD 20850
301-838-5000
http://www.intersolv.com/

Caribou Lake Software Inc.
4780 Beacon Hill Road
St. Paul, MN 55122
612-688-9470
http://www.cariboulake.com/

SCO
425 Encinal Street
P.O. Box 1900
Santa Cruz, CA 95061-1900
408-425-7222
http://www.sco.com/

Sybase Inc.
6475 Christie Avenue
Emeryville, CA 94608
510-922-3555
http://www.sybase.com/

Oracle Corp.
500 Oracle Parkway
Redwood Shores, CA 94065
650-506-7000
http://www.oracle.com/

Connect Software Inc.
81 Lansing Street
San Francisco, CA 94105
415-543-6695
http://www.connectsw.com/

I-Kinetics Inc.
17 New England Executive Park
Burlington, MA 01803
781-270-1300
http://www.i-kinetics.com/

DDJ

OPENGL

Listing One

```
typedef struct
{
     float x, y, z;  /* coordinates */
} make_vertex_list;
```

Listing Two

```
typedef struct
{
    int a, b, c;     /* array indices */
    int a_s, a_t,    /* (s, t) texture coordinates */
    b_s, b_t,
    c_s, c_t;
} make_index_list;
```

Listing Three

(a)

```
(vertex_list[index_list[0].a].x, vertex_list[index_list[0].a].y,
                                 vertex_list[index_list[0].a].z)
```

(b)

```
(vertex_list[index_list[0].b].x, vertex_list[index_list[0].b].y,
                                  vertex_list[index_list[0].b].z)
```

(c)

```
(vertex_list[index_list[0].c].x, vertex_list[index_list[0].c].y,
                                  vertex_list[index_list[0].c].z)
```

Listing Four

```
typedef struct
{
     make_vertex_list *vertex;
} make_frame_list;
```

Listing Five:

(a)

```
frame_list[F].vertex[index_list[P].a].x
frame_list[F].vertex[index_list[P].a].y
frame_list[F].vertex[index_list[P].a].z
```

(b)

```
frame_list[F].vertex[index_list[P].b].x
frame_list[F].vertex[index_list[P].b].y
frame_list[F].vertex[index_list[P].b].z
```

(c)

```
frame_list[F].vertex[index_list[P].c].x
frame_list[F].vertex[index_list[P].c].y
frame_list[F].vertex[index_list[P].c].z
```

Listing Six

(a)

```
R: m_palette_buffer [ m_pixel_buffer[0]]
G: m_palette_buffer [ m_pixel_buffer[1]]
B: m_palette_buffer [ m_pixel_buffer[2]]
```

(b)

```
R: m_palette_buffer [3 * m_pixel_buffer[P]+0]
G: m_palette_buffer [3 * m_pixel_buffer[P]+1]
B: m_palette_buffer [3 * m_pixel_buffer[P]+2]
```

(c)

```
R: m_palette_buffer [3 * m_pixel_buffer[X + Y*Width]+0]
G: m_palette_buffer [3 * m_pixel_buffer[X + Y*Width]+1]
B: m_palette_buffer [3 * m_pixel_buffer[X + Y*Width]+2]
```

Listing Seven

```
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

Listing Eight

```
glEnable(GL_CULL_FACE);
glEnable(GL_TEXTURE_2D);
glPolygonMode (GL_FRONT, GL_FILL);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
```

Listing Nine

```
Glubyte *unScaled = new GLubyte [m_iWidth * m_iHeight * 4];
for (j = 0; j < m_iHeight; j++) {
  for (i = 0; i < m_iWidth; i++) {
    unScaled[4*(j * m_iWidth + i)+0] =
            (GLubyte) m_palette_buffer[3*m_pixel_buffer[j*m_iWidth+i]+0];
    unScaled[4*(j * m_iWidth + i)+1] =
            (GLubyte) m_palette_buffer[3*m_pixel_buffer[j*m_iWidth+i]+1];
    unScaled[4*(j * m_iWidth + i)+2] =
            (GLubyte) m_palette_buffer[3*m_pixel_buffer[j*m_iWidth+i]+2];
    unScaled[4*(j * m_iWidth + i)+3] = (GLubyte) 255;
  }
}
```

Listing Ten

```
/* allocate memory for the new rescaled texture */
glTexture = new GLubyte [m_iscaledWidth * m_iscaledHeight * 4];

/* use the OpenGL function to rescale */
gluScaleImage (GL_RGBA, m_iWidth, m_iHeight, GL_UNSIGNED_BYTE, unScaled,
            m_iscaledWidth, m_iscaledHeight, GL_UNSIGNED_BYTE, glTexture);

/* reclaim memory of the unscaled texture */
delete [] unScaled;
```

TEXTURE MAPPING

Listing One

```
void Draw_Triangle(float x0,float y0,float x1,float y1,
                                     float x2,float y2, int color)
{
// this function rasterizes a triangle with a flat bottom

// compute left side interpolant
float dx_left = (x2 - x0)/(y2 - y0);

// compute right side interpolant
float dx_right = (x1 - x0)/(y2 - y0);

// seed left and right hand interpolators
float x_left = x0;
float x_right = x0;

// enter into rasterization loop
for (int y=y0; y<=y1; y++)
     {
      // draw the scanline
      Draw_Line(x_left, x_right, y, color);

     // advance interpolants
     x_left+=dx_left;
     x_right+=dx_right;

     } // end for y
} // end Draw_Triangle
```

Listing Two

```
// initialize u,v interpolants to left and right side values
ui = ul;
vi = vl;

// now interpolate from left to right, i.e, in a positive x direction
for (x = xstart; x <= xend; x++)
     {
      // get texture pixel value
      pixel = texture_map[ui][vi];

      // plot pixel at x,y
      Plot_Pixel(x,y,pixel);

      // advance u,v interpolants
      ui+=du;
      vi+=dv;
      } // end for x
```

DVD

Listing One

```
// grab an interface to the Annex J methods
hr = m_pgraph->QueryInterface(IID_IDvdControl, (void **) &m_pUserOperations);

if ( !(FAILED(hr)) )
{
    // start playing title 1, chapter 3
    hr = m_pUserOperations->ChapterPlay( 1, 3 );

    // view the 2nd angle
    m_pUserOperations->AngleChange( 2);

    // turn off annoying foreign language subtitles
    m_pUserOperations->SubpictureStreamChange( 1, FALSE );

    // release interface
    m_pUserOperations->Release();
}
```

Listing Two

```
switch (Event)
    {
        case EC_DVD_BUTTON_CHANGE     :
            // lParam1 contains number of active buttons
            // if there are ANY buttons alive--enable button manipulation
            if ( lParam1 > 0 )
            {
                bActiveButtons = TRUE;
            }
            break;
        case EC_DVD_ANGLE_CHANGE      :
                    // lParam1 contains the current viewing angle
            break;
        case EC_DVD_TITLE_CHANGE      :
            // lParam1 contains the currently playing title
            break;
        case EC_DVD_CHAPTER_START     :
            // lParam1 contains the currently playing chapter
            break;
        case EC_DVD_VALID_UOPS_CHANGE :
            // lParam1 contains the currently active user operations
            break;
    }
```